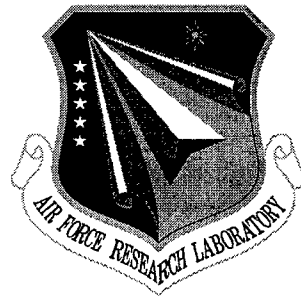


**AFRL-IF-RS-TR-2001-118**  
**Final Technical Report**  
**June 2001**



# **DATA MULTICASTING IN HIGH SPEED MULTI - SERVICE NETWORKS**

**Kansas State University**

**K. Ravindran**

*APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.*

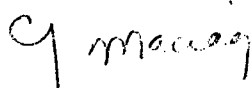
**20010809 032**

**AIR FORCE RESEARCH LABORATORY  
INFORMATION DIRECTORATE  
ROME RESEARCH SITE  
ROME, NEW YORK**

This report has been reviewed by the Air Force Research Laboratory, Information Directorate, Public Affairs Office (IFOIPA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

AFRL-IF-RS-TR-2001-118 has been reviewed and is approved for publication.

APPROVED:



CHESTER J. MACIAG  
Project Engineer

FOR THE DIRECTOR:



WARREN H. DEBANY, Technical Advisor  
Information Grid Division  
Information Directorate

If your address has changed or if you wish to be removed from the Air Force Research Laboratory Rome Research Site mailing list, or if the addressee is no longer employed by your organization, please notify AFRL/IFGB, 525 Brooks Road, Rome, NY 13441-4505. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document require that it be returned.

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.				
1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE JUNE 2001		3. REPORT TYPE AND DATES COVERED Final Aug 94 - Aug 96
4. TITLE AND SUBTITLE DATA MULTICASTING IN HIGH SPEED MULTI-SERVICE NETWORKS			5. FUNDING NUMBERS C - F30602-94-C-0241 PE - 62702F PR - 4519 TA - 22 WU - 37	
6. AUTHOR(S) K. Ravindran				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Kansas State University Department of Computing and Information Sciences Manhattan Kansas 66506			8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Research Laboratory/IFGB 525 Brooks Road Rome New York 13441-4505			10. SPONSORING/MONITORING AGENCY REPORT NUMBER  AFRL-IF-RS-TR-2001-118	
11. SUPPLEMENTARY NOTES Air Force Research Laboratory Project Engineer: Chester J. Maciag/IFGB/(315) 330-3184				
12a. DISTRIBUTION AVAILABILITY STATEMENT APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.			12b. DISTRIBUTION CODE	
13. ABSTRACT (Maximum 200 words) The project focuses on architectural and protocol techniques for multicast transport of data in the evolving applications (such as video, audio, and graphics). The techniques allow creating a 'programmable network' that may be 'plugged - in' with QOS and flow parameters of data streams to instantiate the network behavior for matching the needs of each applications. The 'programmability' of the network can allow reducing the deployment of communication resources (such as bandwidth) for supporting applications. This philosophy is in alignment with the evolving 'Internet Service Layer' functionalities. A canonical network substrate so created may then be employed to construct multicast transport services. A tree - structured channel in the network is used as building block for realizing multicast data transport. Data from different sources are multiplexed over a shared tree channel for reaching destinations through intermediate network nodes and links. The sharing of segments across multiple data streams allows reducing the overall fixed costs of maintaining 'connections' in the backbone network, and also offers the potential for reduced bandwidth allocation for bursty data flows due to 'statistical multiplexing' of these flows. We developed resource allocation protocols and routing algorithms for multicast networks, based on shared tree channels. The protocols were evaluated by 'modeling' and 'experimentation' activities on LANs and the ATM-based testbed NYNET at Rome Laboratory. Applications involving the distribution of video and audio data were developed to demonstrate the viability of the multicast network technology.				
14. SUBJECT TERMS Data Multicasting, High Speed Multi-Service Networks, Programmable Network, Internet Service Layer, Local Area Networks, LANs, ATM-Based Testbed, Networks			15. NUMBER OF PAGES 56	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT  UNCLASSIFIED		18. SECURITY CLASSIFICATION OF THIS PAGE  UNCLASSIFIED		19. SECURITY CLASSIFICATION OF ABSTRACT  UNCLASSIFIED
			20. LIMITATION OF ABSTRACT  UL	

# Contents

<b>Project summary</b>	iv
<b>1 Project motivation and direction</b>	<b>1</b>
<b>2 Proposed architectural elements of transport networks</b>	<b>4</b>
2.1 Distribution tree . . . . .	5
2.2 Paradigms for integration of QOS support in transport networks . .	6
2.3 Layers of transport functions . . . . .	6
<b>3 Shared tree based multicast architectures</b>	<b>8</b>
3.1 Sharing of path segments . . . . .	8
3.2 Architectural support for shared trees . . . . .	9
3.3 Comparative analysis of multicast architectures . . . . .	10
3.4 Empirical studies on path costs . . . . .	12
3.5 Implications on routing algorithms . . . . .	12
<b>4 Network elements for resource control</b>	<b>15</b>
4.1 Switch level resource allocator . . . . .	15
4.2 Switch level QOS controller . . . . .	16
4.3 Stream filters . . . . .	16
<b>5 Design of multicast routing control protocols</b>	<b>18</b>
5.1 Functional decomposition of routing algorithms . . . . .	18
5.2 End-to-end delay constraints . . . . .	20
5.3 Routing control protocol . . . . .	20
5.4 Simulation studies . . . . .	22
<b>6 Implementation study over ATM networks</b>	<b>23</b>
6.1 Routing protocol overlay . . . . .	23
6.2 'native ATM signaling' support . . . . .	24
6.3 Application configurations . . . . .	25
<b>7 Study of multicasting over heterogeneous networks</b>	<b>25</b>
7.1 Semantics of logical addresses . . . . .	26
7.2 Packet forwarding . . . . .	27

7.3	Implementation strategies for IP LANs . . . . .	28
7.4	Implementation strategies for SMDS networks . . . . .	30
<b>8</b>	<b>Development of end-system support mechanisms</b>	<b>32</b>
8.1	Grouping of application-level devices . . . . .	33
8.2	Workstation software for multicast transport . . . . .	34
8.3	Connecting application devices . . . . .	34
		<b>37</b>
<b>9</b>	<b>Technical deliverables</b>	<b>40</b>
9.1	Study of multicast network architectures . . . . .	40
9.2	Design of multicast routing control protocols . . . . .	40
9.3	Design of signaling system overlays on ATM networks . . . . .	40
9.4	Interconnection of heterogeneous backbone networks . . . . .	41
9.5	End-system software on workstations . . . . .	41
9.6	Software demonstration . . . . .	41
<b>10</b>	<b>Project contributions to Air Force research mission</b>	<b>41</b>
<b>11</b>	<b>Future works</b>	<b>43</b>
<b>12</b>	<b>Scientific contributions</b>	<b>44</b>

## List of Figures

Figure 1.	Multi-destination data delivery in a multi-service network	2
Figure 2.	Functional requirements of multicast models and their mapping to target networks	3
Figure 3.	Distribution tree as building block for multicast architectures	5
Figure 4.	Layers of functions in a multicast transport architecture	7
Figure 5.	Sharing of multicast path segments across streams	9
Figure 6.	Illustration of architectural support for path sharing	10
Figure 7.	Sample multicast channel configurations in SSRT, CBT, PIM and URT architectures	13
Figure 8.	Illustration of flow-to-resource mapping tables in nodes	15
Figure 9.	Illustration of end-to-end delay control information in switches	17
Figure 10.	Illustration of network supported filters of data streams	18
Figure 11.	A 'signaling' oriented structure of multicast service interface	18
Figure 12.	Functional components of multicast routing algorithms	19
Figure 13.	Protocol model for multicast path set up	21
Figure 14.	A 'protocol-oriented' view of resource control system	22
Figure 15.	An implementation structure for 'network level signaling' to support resource Allocations in ATM networks	24
Figure 16.	Path addressing in various layers of multicast network architecture	27
Figure 17.	Multicasting across interconnection to IP LAN's	29
Figure 18.	Overlaying URT-based multicast model on SMDS networks	32
Figure 19.	Illustration of feeder plant mechanisms	33
Figure 20.	Multicast end-system software modules in a workstation	35

### Project summary

The project focuses on architectural and protocol techniques for multicast transport of data in the evolving applications (such as video, audio and graphics). The techniques allow creating a 'programmable network' that may be 'plugged-in' with QOS and flow parameters of data streams to instantiate the network behavior for matching the needs of each application. The 'programmability' of the network can allow reducing the deployment of communication resources (such as bandwidth) for supporting applications. This philosophy is in alignment with the evolving 'Internet Service Layer' functionalities. A canonical network substrate so created may then be employed to construct multicast transport services.

A tree-structured channel in the network is used as building block for realizing multicast data transport. Data from different sources are multiplexed over a shared tree channel for reaching destinations through intermediate network nodes and links. The sharing of segments across multiple data streams allows reducing the overall fixed costs of maintaining 'connections' in the backbone network, and also offers the potential for reduced bandwidth allocation for bursty data flows due to 'statistical multiplexing' of these flows.

We developed resource allocation protocols and routing algorithms for multicast networks, based on shared tree channels. The protocols were evaluated by 'modeling' and 'experimentation' activities on LANs and the ATM-based testbed NYNET at Rome Laboratory. Applications involving the distribution of video and audio data were developed to demonstrate the viability of the multicast network technology.

Major uses of the project will be in 'Integrated Service Networks' that are becoming the basis for **Distributed Information Environments**: in both military and civilian applications.

# 1 Project motivation and direction

There has been an increasing need to deliver digital subscriber services of diverse characteristics and features to subscriber terminals (e.g., digital TV, video conferencing, distributed computing) through a wide-area and/or a metropolitan area network. Applications involving the distribution of visual still images, video sequences and animated scientific visualizations have begun to dominate the transport load offerings on the network. One of the functional capabilities required of such a *multi-service* network is data multicasting, i.e., dispatching a data on one or more links of a network node (or switch), for multi-destination delivery. The multicast capability should be provided at multi-megabit data rates, such as 10-15 *mb/sec* for compressed video and 100 *mb/sec* for composite imaging. Furthermore, the data transport requirements of the evolving applications are becoming more diverse, expecting different levels of data delivery guarantees from the network. Such requirements, often prescribed as a *quality of service* (QOS), include timely delivery of data such as video and audio, and a guaranteed bandwidth to sustain a certain transfer rate such as live video images. See Figure 1. The multicasting requires *network level support* whereby the data from a source entity are transported through multiple network paths towards destination entities, while meeting the application level QOS requirements.

## Requirements of multicast networks

In light of the evolution of diverse applications on one hand and that of backbone network technologies (such as 'ATM cell switching' based fiber networks, interconnected 'packet switching' LANs/wireless networks and high level 'frame switching' networks) on the other hand, it is essential that the support of multicast functions by a switching system meets the following requirements:

- Provide uniform set of mechanisms to transport the data of diverse applications over a backbone network;
- Allow scalable implementation of applications on a variety of backbone networks;
- Be usable in an environment where heterogeneous backbone networks may co-exist;
- Be extensible to allow smooth introduction of newer applications and backbone network technologies.



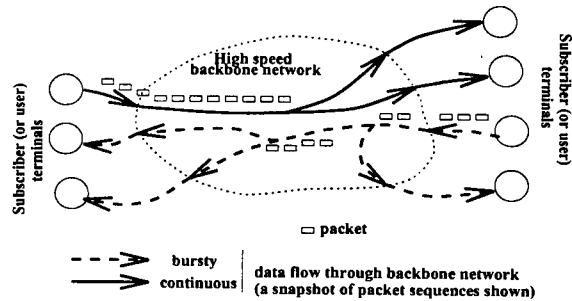


Figure 1: Multi-destination data delivery in a multi-service network

In this R & D contract undertaken for the Rome Laboratory, we investigated the issues in providing generic multicast models to meet the above requirements and generating flexible transport architectures that incorporate the models. These architectures are amenable for realization on a variety of high speed backbone networks.

### Are existing communication models suitable ??

There are three aspects of a given multicast model that determine the flexibility, extensibility and scalability of networks implemented using the model to support multi-service applications (see Figure 2):

- Richness of multicast service interface provided by the model

Typically, the multicast interface should allow the user entities to specify transport attributes of data, viz., quality of transport service (QOS), such as transfer rates, burstiness and acceptable end-to-end delays for any type of application in a uniform manner. It should also allow any arbitrary set of destinations to be selected for data delivery (e.g., private discussion groups in a video conference).

- Adaptability of architectural components of the model to backbone networks

The communication architecture of the model should be embeddable onto any type of target backbone network. To allow this, the layer functions and the inter-layer boundaries in the architecture should be well-specified so that they can be appropriately mapped to the target network layers. For instance, how a global multicast supported by the model interworks with the native multicast facility available in backbone networks (e.g., LANs) should be concretely specified.

- Compatibility of protocol elements supported by the model

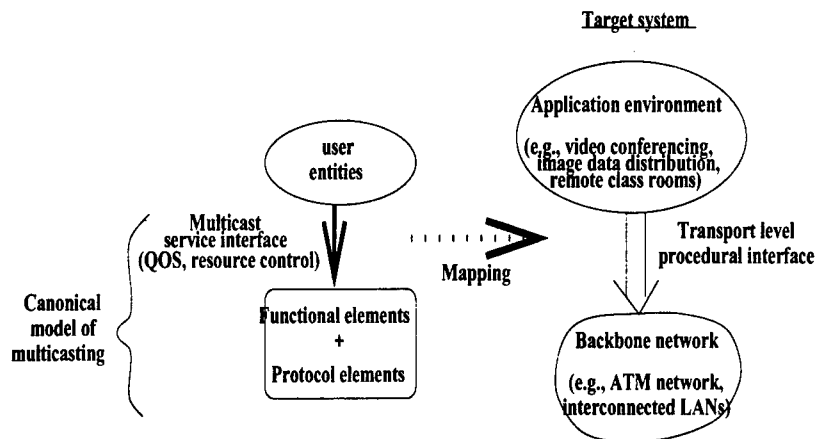


Figure 2: Functional requirements of multicast models and their mapping to target networks

The protocol elements used to realize the model should be mappable onto the protocols supported in the target network. For instance, the ‘source-routing’ of packets as allowed in the model (say) should be supported by a ‘backbone adaptation layer’ that allows variable length packet headers.

A model that is not comprehensive enough to support all these aspects is deemed restrictive in its usability in implementing multi-service networks.

Many network models that evolved in the ARPANET, OSI and telephone network worlds do not meet the above requirements. For instance, a packet transport over point-to-point paths set up between each destination and the source as allowed by OSI network layer standards [1] may cause a packet to be sent more than once over a link shared between the various paths, thereby entailing wastage of link bandwidth. Likewise, the IP ‘host group’ model for internet multicasting [2] is usable only in IP based networks. Though Deering’s earlier work on multicasting provides many features [3], it does not allow user level specification of QOS to be supported by the network. We shall provide more concrete comparisons with existing models later in the report.

Thus it is necessary to provide a comprehensive model of multicasting meeting all the requirements stated earlier.

### Focus of project activities

We adopted the ‘proof-of-concept by modeling and experimentation’ to substantiate the key ideas in the project. The ‘modeling’ component of work involved the generation of canonical network architectures and service paradigms, and develop-

ment of simple prototypes of these architectures on a small Ethernet LAN-based testbed at Kansas State University. The 'experimentation' component of the project involved implementing the models on the ATM-based NYNET testbed at the Rome Laboratory. The 2-tier approach separated the 'modeling' and 'experimentation' components, with the concepts generated by the former validated by the latter.

The **Statement of Work** phrases attached to the contract with Rome Laboratory were used to set the 'direction' of this project. The 'statement of work', in a larger context, directs the project towards developing a technology to support a **Distributed Information Environment** for use by civilian and/or military applications.

### Organization of report

Section 2 identifies the basic ingredients required of multicast transport architectures for supporting multi-service applications. Using these requirements as guidelines, section 3 describes an architecture developed in this project to support QOS and flow specifications, and compares it with currently available architectures. Section 4 describes the canonical functional elements incorporatable into the various architectures for multicast transport. Section 5 describes the design of multicast routing control protocols for use in these architectures. Section 6 describes a case study of architecture implementation over the ATM-based testbed to support multicast-based applications (such as multimedia conferencing). Section 7 describes the strategies for implementing multicast on various backbone networks. Section 8 describes the multicast support mechanisms we developed for end-system stations. Sections 9-10 highlight the 'technical deliverables' of this project and its relevance to Air Force Research Missions. Sections 11-12 conclude the report by identifying the future works and research activities spawned off from this project.

## **2 Proposed architectural elements of transport networks**

Our basic premise is that a multicast transport architecture should be *network-centric* in that the user-network interface should allow casting the needs of applications onto network internal mechanisms. For example, a path leading to audio-only terminals may be set up over a 100 *kb/sec* link, as against a path leading to video+audio terminals set up over a 3 *mb/sec* link. In such an architecture, the user-level flow specifications are transcribed to the network for the latter to determine an appropriate data distribution path that minimizes the deployment of communication

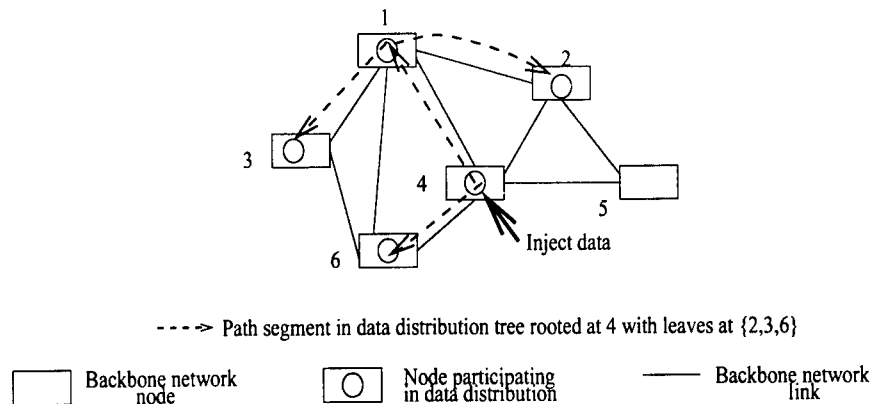


Figure 3: Distribution tree as building block for multicast architectures

resources, viz., link bandwidth and router node buffers.

In this section, we describe the basic elements of multicast architectures for supporting the QOS specifications.

## 2.1 Distribution tree

A multicast tree represents a logical topology superimposed on backbone network nodes and links such that a data can be injected at root node of the tree and get replicated along intermediate branches of the tree for reaching destinations at leaf nodes of the tree [3, 4]. See Figure 3. Since a tree topology does not have cycles, the forwarding of a data unit received over the input link of a node is only a matter of determining the output links to send the data unit. The data from various sources are made available at the root node of a tree through one or more point-to-point paths for forwarding towards destinations.

Each node in a multicast path implements: i) a *routing function* that determines the outgoing links for an incoming data, and ii) a *resource allocation* function that reserves node buffers and link bandwidths to sustain the required data flow. The setting up of tree-structured path and point-to-point paths spanning the sources and destinations in an application is carried out by a multicast routing algorithm. The path determination may be based on the total amount of resource consumptions and routing overhead in getting the data flows from sources to destinations through the backbone network (which constitutes the transport cost).

## 2.2 Paradigms for integration of QOS support in transport networks

In determining an appropriate architecture, we employ 'programmable network' as the underlying theme that is becoming the basis for building large networked systems, such as the Internet. There are somewhat different schools of thought on the notion of 'programmable network':

*Service-oriented view* [5]: A network model that can offer a variety of service classes for data delivery such as delay jitter-free and jitter-allowed data transport for high-fidelity audio and text based conferencing respectively;

*Resource allocation-oriented view* [6]: A network model that allows users to define 'data flows' and their requirements which can be passed on to network subsystems for resource allocations, viz., bandwidth, buffers and state in the router nodes;

*Network-independent bearer service view* [7]: An abstract organization of protocols and service interfaces not tied to any existing protocol suite, that is capable of cross-connecting applications to backbone networks.

We believe that the above views are intricately inter-woven with one another, and hence need to be taken into account in the design of any large network system. Accordingly, our formulation of tree-based multicast transport architectures is guided by these views.

A concretization of the 'programmable network' view in transport architectures manifests in the following aspects:

- Fine granular control on network resource allocations through application level QOS/flow specifications;
- Extensibility of allowing QOS/flow based 'network parameterization' for diverse applications.

Incorporating these aspects, we formulated canonical models/architectures for multicasting, and then mapped them to operational backbone networks (such as ATM networks). Refer to Figure 2.

## 2.3 Layers of transport functions

The source and destination entities involved in a multicast data transport may be viewed as organized in a flat group (UTL). These entities exercise multicast control

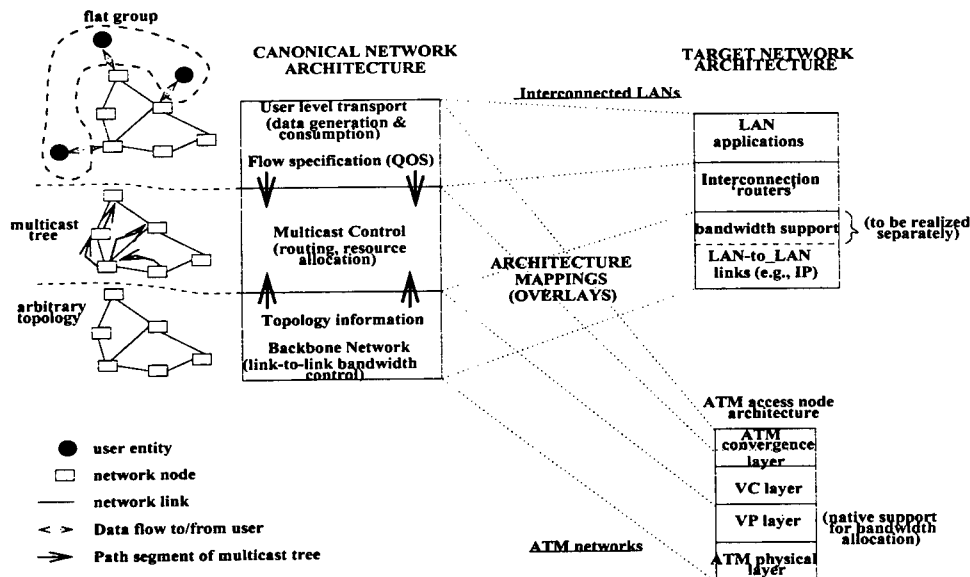


Figure 4: Layers of functions in a multicast transport architecture

functions (MCL) to set up a tree-structured distribution path over the physical topology of backbone network nodes and links that can carry the required data flows from sources to destinations. The backbone network (BNL) is assumed to provide node-to-node link bandwidth reservation upon demand.

The UTL, MCL and BNL functions are mappable onto appropriate layers in a target network architecture. The mapping may involve the canonical functions in these layers to be bound to the local functions in the target network. See Figure 4. In interconnected LANs for instance, the UTL functions reside in LAN applications, MCL functions in interconnection routers, and BNL functions in LAN link-to-link connectivity layer augmented with bandwidth allocation control. In ATM networks for instance, the BNL functions map to the 'virtual path' (VP) layer with 'native' bandwidth allocation support, and the MCL functions are placed in the 'virtual circuit' (VC) layer and partly in the convergence layer above, exercising the VP level multicast provided in ATM networks (see section 6). Appropriate convergence sublayers may be employed in target networks to support these mappings.

Employing the tree-structured channels and the flow & QOS support paradigms as building blocks of multicast transport architectures, we now describe the various activities carried out in the project.

### 3 Shared tree based multicast architectures

In this section, we describe the architectural support mechanisms for multi-source data flows in an application. These mechanisms are useful for multipoint-to-multipoint communications where each user entity in an application can be a source of data as well as a destination for the data from other sources. A major application is multimedia conferencing.

#### 3.1 Sharing of path segments

The data stream from each source flows over a tree towards a common set of destinations. The sharing of a tree segment, i.e., backbone network 'connection', across different data streams ('stream multiplexing' over a path segment) allows:

- Amortization of the 'connection' fixed costs (e.g., routing table entries and 'connection' descriptors) across the various streams;
- Better use of 'connection' bandwidth by exercising 'statistical interleaving' of streams along this segment and its downstream path segments, particularly when data flows are bursty (such as compressed video and text);

in comparison to sending each stream over a separate 'connection' set up over backbone network link. See Figure 5 that shows flows 1 and 2 in an application with rates  $q_1$  and  $q_2$  respectively. Empirically, the cost of these flows when multiplexed over a single 'connection' set up over a link may be given by:

$$\begin{aligned} C_{mux} &= fc + h.(q_1 + q_2) \quad \text{when } q_1, q_2 \text{ are average rates} \\ &= fc + h.(q_1 + q_2)^k \quad \text{when } q_1, q_2 \text{ are peak rates (for bursty flows),} \end{aligned} \quad (1)$$

where  $fc$  is the fixed cost of using the 'connection',  $0 \ll k \leq 1.0$  such that data loss does not exceed an application-specified limit  $\Delta$ , and  $h$  is a constant. The equation (1) can be generalized to any number of flows. Here, for a given  $\Delta$ ,  $k$  decreases as the number of streams multiplexed, data burstiness and flow rate are increased. And for a given set of flow parameters,  $k$  decreases as  $\Delta$  is increased. Our experiments on ATM network traffic using 3 bursty streams of 15 *mbps* each with 'burst factor' of 0.5 show that  $k = [0.9, 1.0]$  for  $\Delta = 2\%$  (e.g., voice data can tolerate up to 2% loss without noticeable degradation in quality). When the network cannot relate the flows as belonging to a single application, they may be sent over separate 'connections' set up over the link. Here, the cost of flows 1 and 2 may be given by

$$C_{no\_mux} = 2.fc + h.(q_1 + q_2) \quad \text{when } q_1, q_2 \text{ are average rates}$$

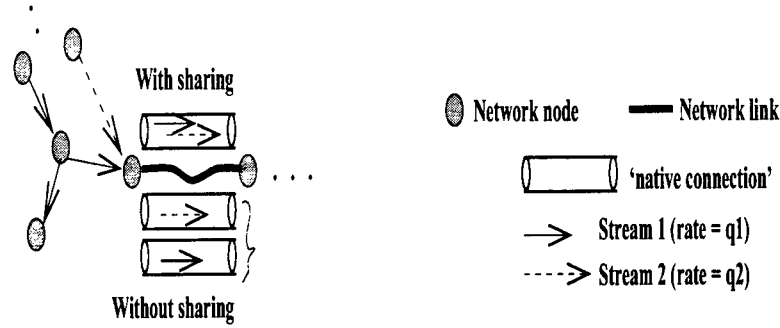


Figure 5: Sharing of multicast path segments across streams

$$= 2.fc + h.(q_1 + q_2)^{k'} \quad \text{when } q_1, q_2 \text{ are peak rates (for bursty flows)} \quad (2)$$

where  $k \leq k' \leq 1.0$ . The parameter  $k'$  takes into account any 'statistical interleaving' of data units over the link, as supported internally by the backbone network (e.g., the multiplexing of 'cells' of different 'virtual circuits' over 'virtual path' links in ATM networks, based on network-assigned 'bandwidth classes').

As can be seen,  $C_{no\_mux} > C_{mux}$ . The less savings in bandwidth allocation arise because the network does not have information that will enable it to determine the criteria for 'statistical interleaving' of streams with respect to  $\Delta$ . Thus, sharing of path segments across data streams reduces the per-stream fixed costs, and offers the potential for reducing the flow costs.

The capability to share tree channels across data streams is itself specific to a network architecture for multicasting.

### 3.2 Architectural support for shared trees

Here, 'sharing' means that when a stream gets combined with another stream, both the streams are routed through common links all the way downstream towards destinations.

A key feature of multicast architectures is the set of nodes in physical topology of backbone network that are allowed to be chosen as *stream multiplexing points* (SMP). The data from various sources flow to a SMP node, and then the combined data flows along a tree rooted at this SMP node towards destinations. See Figure 6. A routing algorithm determines the data flow paths so as to optimize the network-wide cost of data transport. Thus, for a source in node  $s$  and destinations in nodes  $\{d\}$ , the transport cost may be given as:

$$net\_cost(s, \{d\}) = flow\_cost(s, [s, SMP(s)]) + mux\_flow\_cost(s, [tr(SMP(s), \{d\})]) +$$



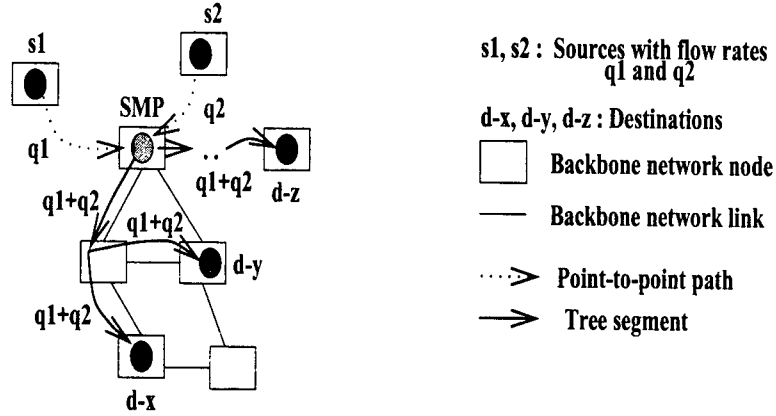


Figure 6: Illustration of architectural support for path sharing

$$fixed\_cost([s, SMP(s)]) + fixed\_cost([tr(SMP(s), \{d\})]), \quad (3)$$

where  $[s, SMP(s)]$  refers to a point-to-point path from  $s$  to  $SMP(s)$ , and  $[tr(SMP(s), \{d\})]$  refers to the tree with root at  $SMP(s)$  and leaves at  $\{d\}$ . Both  $mux\_flow\_cost$  and  $flow\_cost$  depend on the number of hops in a path and the bandwidth needs of flow from  $s$ ; the  $mux\_flow\_cost$  also depends on the number of streams multiplexed with  $s$  and the burstiness of various streams; the  $fixed\_cost$  depends on the number of hops in a path,  $fc$ , and the number of streams multiplexed. In general, more the number of streams multiplexed along a path, lower the  $net\_cost$ .

We now analyze the extent of support for shared distribution paths in current multicast architectures.

### 3.3 Comparative analysis of multicast architectures

Let  $V$  be the set of nodes in physical topology of the backbone network, and  $\{s_i\}_{i=1,2,\dots,N}, \{d_j\}_{j=1,2,\dots,M} \subseteq V$  be the nodes containing source and destination entities respectively. For a given placement of  $\{s_i\}$  and  $\{d_j\}$  in the topology, we examine the cost of multicast paths in various architectures.

#### 'source-rooted tree' (SSRT) architecture

The stream from each source flows over a separate tree that serves all destinations. There is no sharing of the common links across various streams. So the transport cost is given by

$$net\_cost(\{s_i\}, \{d_j\}) = \sum_{\forall i} fixed\_cost([tr(s_i, \{d_j\})]) + flow\_cost(s_i, [tr(s_i, \{d_j\})]).$$

The 'Internet Stream Protocol' (ST-II) developed at BBN [8] and the 'distance vector multicast routing protocol' [9] fall under this architecture.

### 'core-based tree' (CBT) architecture [10]

The streams from all sources are first collected at a fixed central node, called the 'core'; the combined streams are then sent over a distribution tree rooted at the 'core' and serving all destinations. Here, any node in the tree that is directly reachable from a source can serve as its SMP. The path segments from the 'core' to destinations are shared across streams, but the point-to-point paths from sources to the 'core' are not. Accordingly, the transport cost is given by

$$net\_cost(\{s_i\}, \{d_j\}) = \sum_{v_i} (flow\_cost(s_i, [s_i, core]) + fixed\_cost([s_i, core])) + fixed\_cost([tr(core, \{d_j\})]) + \sum_{v_i} mux\_flow\_cost(s_i, [tr(core, \{d_j\})]).$$

### 'rendezvous-point rooted tree' (PIM) architecture [11]

The streams from all sources are first collected at a fixed set of nodes, called the 'rendezvous points' (RP); the combined streams are then sent over the distribution trees rooted at these RPs and serving each a disjoint subset of destinations. A source connects to each of the RP nodes separately, to serve as its SMP for destinations connected by the tree therein. The path segments from a RP to its subset of destinations are shared across streams, but the point-to-point paths from sources to the RP are not. Accordingly, the transport cost is given by

$$net\_cost(\{s_i\}, \{d_j\}) = \sum_{x \in RP} \left( \sum_{v_i} (flow\_cost(s_i, [s_i, x]) + fixed\_cost([s_i, x])) + fixed\_cost([tr(x, \{d_j(x)\})]) + \sum_{v_i} mux\_flow\_cost(s_i, [tr(x, \{d_j(x)\})]) \right),$$

where  $d_j(x)$  is a destination served by the tree rooted at RP  $x$ .

### 'unrooted tree' (URT) architecture [12, 13, 14]

The streams from sources may be collected at any set of nodes; the combined streams are then sent over the distribution trees rooted at these nodes and serving various destinations. Here, any node in the channel that is directly reachable from a source can serve as its SMP. In contrast to CBT and PIM architectures, the point-to-point path segments connecting a source to its SMP node are sharable with other sources<sup>1</sup>. Accordingly, the transport cost is given by

$$net\_cost(\{s_i\}, \{d_j\}) = fixed\_cost([G]) + \sum_{v_i} (flow\_cost(s_i, [s_i, SMP(s_i)]) + mux\_flow\_cost(s_i, [tr(SMP(s_i), \{d_j\})]))$$

<sup>1</sup>The SMP function in a node  $x$  for a source  $s$  can dynamically shift to an upstream node  $y$  in the point-to-point path towards  $s$  where a new source  $s'$  connects to. Upon this,  $y$  becomes the SMP for both  $s$  and  $s'$ , and the point-to-point path from  $y$  to  $x$  gets shared by  $s$  and  $s'$ .

where  $\mathcal{G}$  is an acyclic graph such that  $tr(s_i, \{d_j\}) \subseteq \mathcal{G}|_{v_i}$  and  $\mathcal{G} = \bigcup_{v_i} tr(s_i, \{d_j\})$ .

Figure 7 illustrates the multicast paths for a sample source-destination configuration under the SSRT, CBT, PIM and URT architectures each.

### 3.4 Empirical studies on path costs

We have studied a variety of source-destination configurations in physical topologies to determine the network-wide cost savings due to the sharing of path segments across various data streams. With ‘statistical multiplexing’ of bursty streams, bandwidth savings of 18-22% are possible across 5 streams with a burst factor of 0.25 each and  $\Delta = 2\%$ , for a configuration consisting of 8 destinations on a 25-node topology. The savings are higher at about 32% with a burst factor of 0.5 and  $\Delta = 5\%$ . The savings in fixed cost can be as high as 60% in large sized configurations (say, 5 sources and 15 destinations on a 50-node topology); this savings, however, may be perceivable only when the flow costs are relatively small.

Since the degree of path sharing is influenced by the control architecture employed, a cost comparison of multicast transport under different architectures can reveal the extent of savings in resources possible with respect to the base case where no sharing is allowed, as in SSRT (here, the fixed cost overhead is  $\mathcal{O}(N)$ ). In many cases, the degree of sharing and the total number of hops in a multicast path counter-balance each other. For instance, an attempt to achieve the highest degree of sharing often leads the streams to a single SMP at the ‘geographic center’ of the configuration for onward flow, which, however, may increase the number of hops the streams need to traverse. Here, the savings in per-hop bandwidth and/or fixed overheads accrued from the increased sharing of path segments can be out-weighed by the increased ‘path distance’.

### 3.5 Implications on routing algorithms

With the incorporation of QOS-based flow management into routing algorithms, a path with the minimum number of hops for a given flow  $L$  — designated as the ‘shortest path’ in current multicast routers — may not always be the path of choice for  $L$ , due to the following reasons:

- One or more hops in the path not having adequate resources and/or not guaranteeing the delay constraints for  $L$ ;

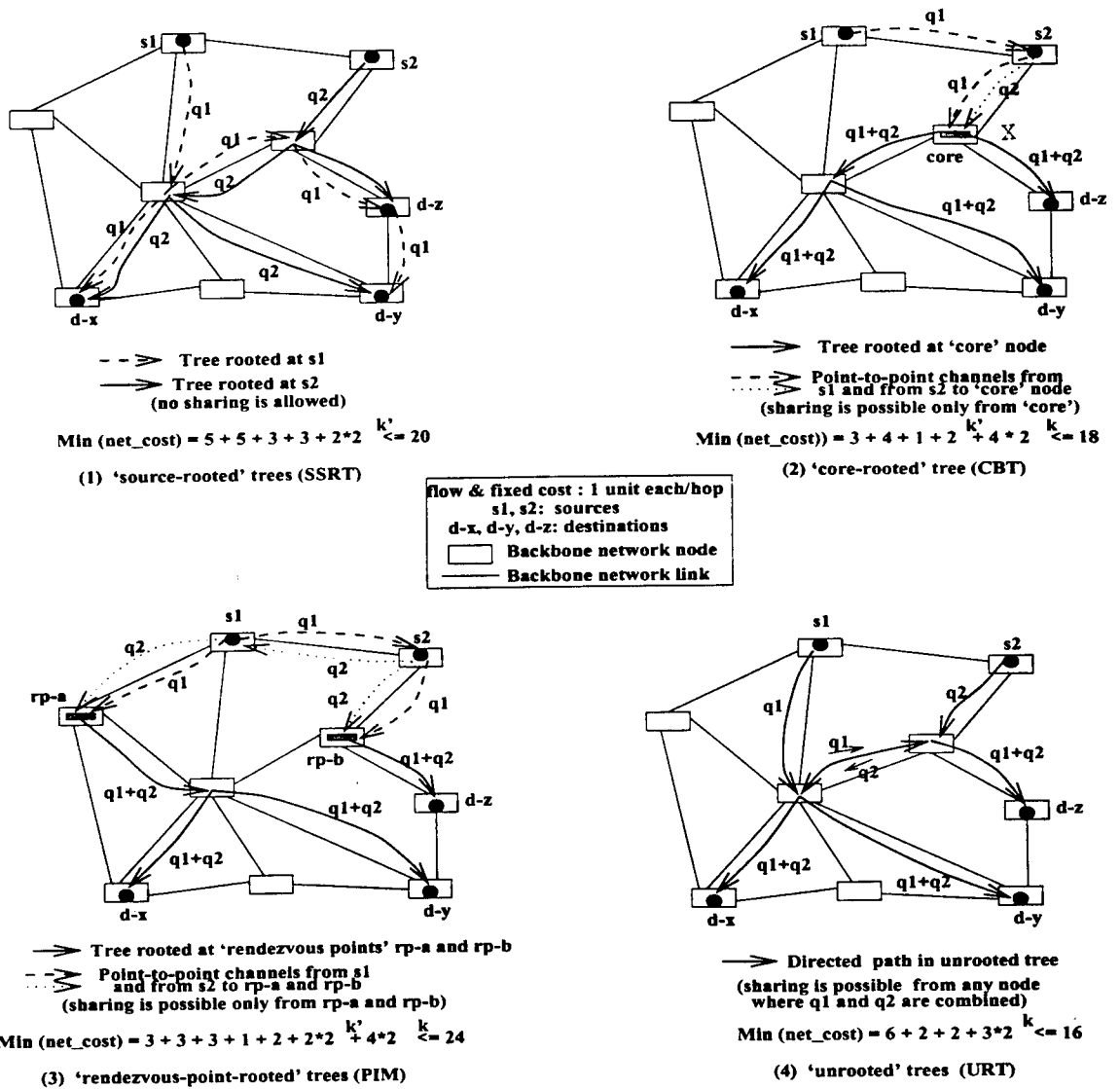


Figure 7: Sample multicast channel configurations in SSRT, CBT, PIM and URT architectures

- A longer path may offer a higher degree of ‘path sharing’ with other flows, resulting in cost savings that may more than offset the cost of sending  $L$  over additional hops<sup>2</sup>.

Thus determining a cost-optimal multicast path is a computationally complex task.

The sharing of distribution paths across multiple streams casts itself upon routing algorithms in the form of: i) constraints on the choice of nodes in a distribution path, ii) the extent of routing control actions required during path setup, and iii) adaptivity to dynamic changes in source-destination configuration and flow parameters. The actual resource consumptions and cost incurred for multicast paths depend on the flow specifications and the source-destination configuration in physical topology.

In SSRT model, each source-rooted tree can be individually minimized, but these non-sharable trees together do not always yield minimality for the entire source-destination configuration. In CBT model, cost minimization of the ‘core-rooted’ tree and the non-sharable point-to-point paths from various sources to ‘core’ node together need not be the best. This is because of the need to send all data to the ‘core’, which limits the ‘spread-out’ of channel topology towards paths of lower cost. In Figure 7 for instance, an optimal placement of ‘core’ node in  $X$  for the sources  $s_1, s_2$  and destinations  $d_x, d_y, d_z$  may not yield cost minimality any more when, say, one of the destinations leaves or a new source/destination joins the configuration. A similar argument holds for PIM model with respect to ‘rendezvous-point’ nodes. With URT model, the trees projected at various sources can be analyzed together for shared path segments, and can be ‘spread out’ with less constraints towards lower cost paths. Thus the cost minimality achievable in various models can be different. See [14, 15] for detailed comparison of various architectures from transport cost and routing algorithm perspectives.

Our treatise on network support for shared distribution paths allows one to determine how far the application-level flow/QOS specifications can be taken down into the

---

<sup>2</sup> An example of analogy is the ‘ride-share’ used by commuters traveling to workplaces in big cities (such as Los Angeles and New York City). Consider 2 commuters  $x$  and  $y$  traveling to a workplace  $z$ . They may ride in separate cars to a ‘park-and-ride’ lot  $r$  and travel therefrom in a single car to  $z$ . In many cases, this may be more cost-effective than  $x$  and  $y$  riding in separate cars each all the way up to  $z$ . This may be the case even if the total distance traveled to  $r$  and then on to  $z$  is higher for  $x$  and/or  $y$  individually. Here, the notion of cost-effectiveness is based on:

- The amortization of fixed costs in using a car, viz., the ‘mechanical wear-and-tear’, ‘driver fatigue’ and ‘highway tolls’, across riders  $x$  and  $y$ ;
- Less amount of variable costs, viz., ‘gasoline consumption’, by a single car with the 2 riders  $x$  and  $y$ , in comparison to that by separate cars for  $x$  and  $y$  each combined.

The use of ‘ride-share’ is subject to the constraint that any increased travel time of  $x$  and  $y$  is less than their ‘tolerable commuting delay’.

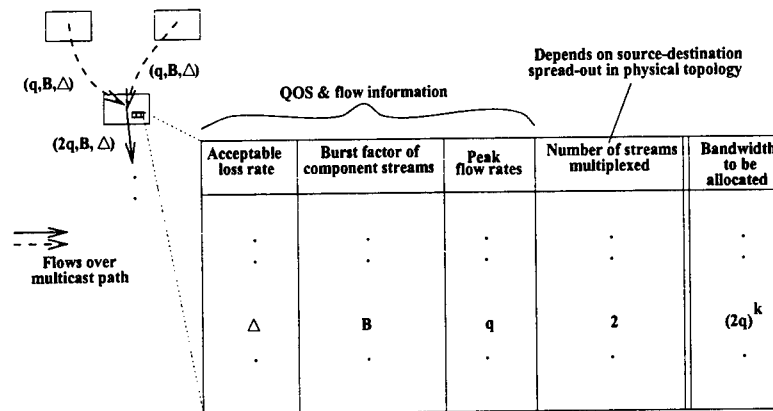


Figure 8: Illustration of flow-to-resource mapping tables in nodes

network to influence its behavior (i.e., ‘network programmability’) and the tradeoffs involved therein.

## 4 Network elements for resource control

In a canonical form, multicast support in the network requires flow/QOS based functional elements that can be appropriately exercised for resource allocation and routing control purposes.

### 4.1 Switch level resource allocator

A switch  $S$  may employ a functional relation  $\mathcal{F}$  that maps a flow rate to resources, viz., link bandwidth needed to support the flow rate and switch buffers needed for sending and/or receiving data through a port at this rate.  $S$  may realize  $\mathcal{F}$  in the form of pre-computed tables that relate the number of streams multiplexed, their average flow rates and burstiness, and the application-wide parameter  $\Delta$  to the bandwidth/buffer needs. See Figure 8 for an illustration.

A specific implementation of these resources, such as the processing cycles consumed in  $S$  to support a flow, is itself a switch internal issue that does not concern a routing control protocol. However, the quantification of resource needs for a given flow has a global scope so that the network-wide cost of a path through  $S$  can be estimated. In other words,  $S$  should support a quantifiable abstraction of resources in order to interwork with a routing algorithm. The de-lination of estimating the resource needs for a flow and a local implementation of resources in switches should be guaranteed by the BNL.

For this purpose, a backbone network may implement a convergence sublayer to export a global view of resources, hiding their local implementation. The global view is then integratable into a MCL function.

## 4.2 Switch level QOS controller

With a 'multicast tree' sharable by more than one data stream, the ability to distinguish the various data streams may still be needed in the network to handle the QOS related transport attributes (say, to map the 'loss sensitivity' and 'acceptable delay jitter' of various data to scheduling priorities on data). For this purpose, the various flow related parameters specified by users are keyed on stream ids.

An important QOS parameter is the acceptable end-to-end delay of data units, i.e., the maximum allowable delay between when a data unit is generated by a source and when it is consumed by a destination. Since each hop in a path segment can add a certain amount of data transfer delay, the end-to-end delay constraint may influence the setting up of tree connecting various destinations. It is also likely that destinations will be able to determine the level of acceptable end-to-end delay on a data stream. In a multimedia conferencing application involving audio and video streams for example, an interactive participant may specify 50 *msec* as the end-to-end delay, while a passive participant may specify 250 *msec*. This receiver-specific delay control allows more flexibility in path selections by a routing algorithm.

Each node  $u$  of a tree maintains information on the cumulative delays incurred by the various streams  $\{s\}$  flowing through  $u$ , denoted as  $\{str\_del(s, u)\}$ . For this purpose,  $u$  may obtain the hop delay information (from BNL) for its incoming path segments from various upstream neighbors (there can be more than one upstream neighbor in URT and CBT architectures, and exactly one upstream neighbor in PIM architecture). A candidate path that connects a destination  $d$  to a source  $s$  through node  $u$  in the tree should have the sum of delays from  $s$  to  $u$  and from  $u$  to  $d$  less than the prescribed end-to-end delay limit  $del_q(s, d)$ . See Figure 9 for an illustration. From among such candidate paths, the routing algorithms chooses a path that minimizes  $net\_cost(s, \{d\})$ .

## 4.3 Stream filters

A source-selective receipt of data by destinations (e.g., an audio-only-capable terminal participating in a video+audio conference session) may be supported in the network by filtering of data streams at various nodes of a shared tree. The filter prevents a stream from flowing along path segments that lead only to destinations

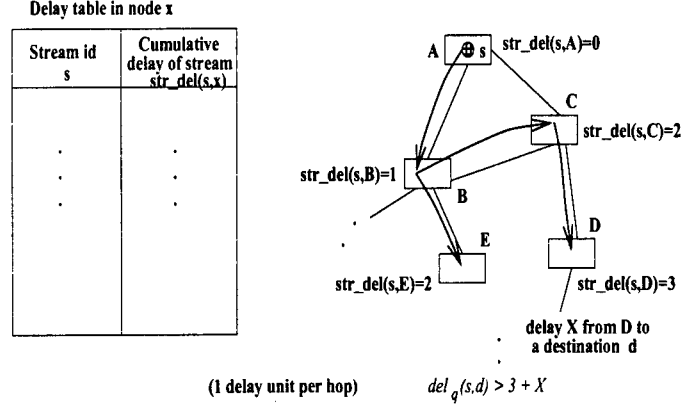


Figure 9: Illustration of end-to-end delay control information in switches

not requiring this stream, thereby saving resources along these path segments. This functional element is somewhat similar to the ‘upward pruning of distribution trees’ proposed in RSVP [16].

A destination  $d$  which *does not* wish to receive data from  $s_1, s_2, \dots, s_m | m \leq N$ , includes the list of stream (id,rate) pairs  $\{(s_j, q_j)\}_{j=1,2,\dots,m}$ , referred to as ‘source mask’  $src\_msk(d)$ , as part of the flow attributes it specifies, where  $N$  is the number of sources in the application. The network splits the combined data stream flowing through the tree at an appropriate node  $u$  to isolate a stream not specified in  $src\_msk$ , for downward flow from  $u$  towards  $d$ . This is done with a filter installed along a path  $e$  of  $u$  through which  $d$  is reachable. The filter maintains a mask list  $k\_lst(u, e) = \{(s_j, q_j)\}$ , and forwards only the data units carrying a stream id not listed in  $k\_lst(u, e)$  through  $e$ . Note that when no filtering is required,  $k\_lst(u, e) = \emptyset$ .

Given that  $d$  does not wish to receive data from a source  $s_i$ , the network may install a filter at each node in the upstream path segment up to the node that roots a subtree with more than one branch and the data is needed along at least one of these other branches. The filter installation also involves de-allocating resources to the extent of  $\mathcal{F}(q_i)$  at each node in this upstream path segment. Thus the flow from  $s_i$  is ON if at least one destination wishes to receive from  $s_i$ . When none of the destinations wish to receive from  $s_i$ , the filter mechanism automatically turns OFF  $s_i$ . See Figure 10 for illustration.

The network functional elements described in this section are exercisable by user level primitives, invoked through a multicast service interface (MSI). See Figure 11. The interactions between the functional elements may be modeled using an ‘object-oriented’ paradigm, which allows these elements to be easily incorporated into ‘resource signaling’ systems.



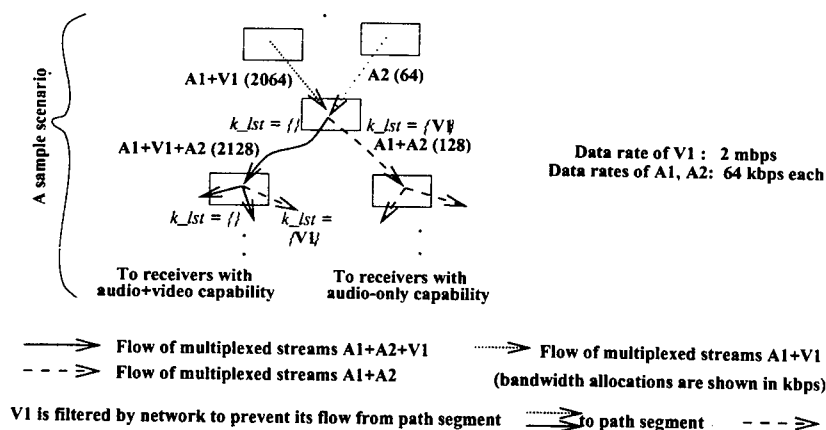


Figure 10: Illustration of network supported filters of data streams

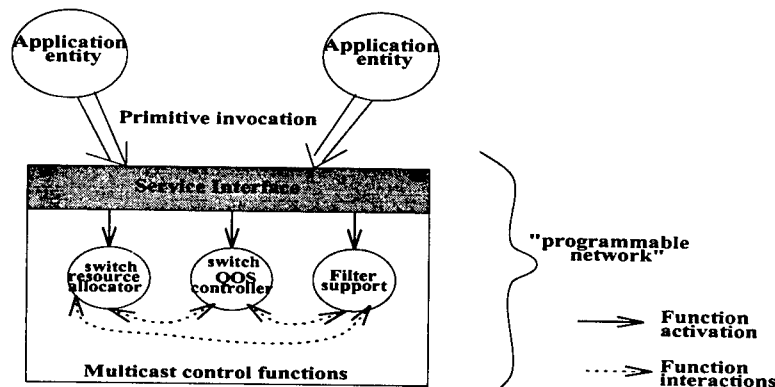


Figure 11: A 'signaling'-oriented structure of multicast service interface

We now describe routing control protocols that allow exercising the network functional elements when user entities connect/disconnect to/from a multicast channel.

## 5 Design of multicast routing control protocols

The protocol support for multicast consists of: i) decentralized information structures maintained at various nodes to support network functional elements, and ii) control message space required to exchange these information across nodes. We first present a high level view of routing algorithms in terms of this protocol support.

### 5.1 Functional decomposition of routing algorithms

A multicast routing algorithm may be viewed as consisting of the following components (see Figure 12):

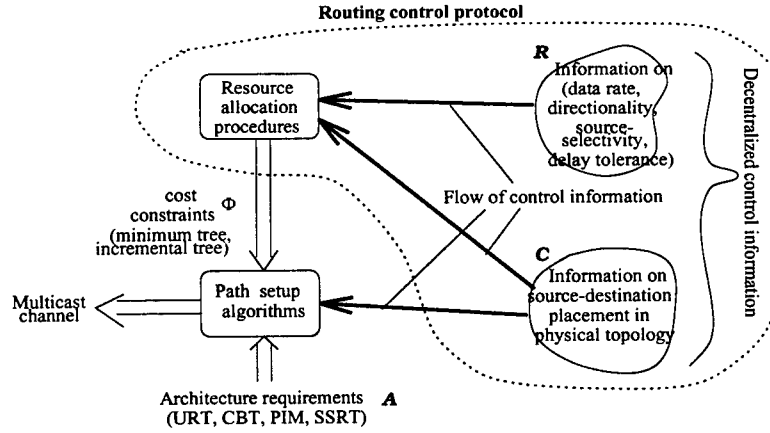


Figure 12: Functional components of multicast routing algorithms

- Mapping of user-level flow specifications  $\mathcal{R}$  to resource demands at a node/link;
- Determining the network-wide resource needs of various interconnection paths for a given source-destination configuration  $\mathcal{C}$ ;
- Optimizing a cost index  $\Phi$  in selecting the data paths;
- Enforcing the architectural constraints  $\mathcal{A}$  on path selection.

Thus a routing algorithm is representable as  $\Phi(\mathcal{F}(\mathcal{R}), \mathcal{C}, \mathcal{A})$ .

A routing algorithm obtains global knowledge of: i) data flows network-wide from various sources to destinations (i.e.,  $\mathcal{R}$ ) based on user level specifications of data directionality, data rates and source-selectivity, and ii) the configuration information (i.e.,  $\mathcal{C}$ ) based on relative placement of source and destination entities in physical topology of backbone network. With the above information, the costs of various possible data paths between sources and destinations under the chosen architecture (i.e.,  $\mathcal{A}$ ) may be computed so that a path meeting a cost and delay constraint (i.e.,  $\Phi$ ) is determined. In the presence of dynamic changes in the configuration,  $\Phi$  may refer to, for instance, a 'network-wide minimal cost' path across every change and a 'incrementally minimal cost' path at each change.

See [4, 17, 18] for representative cost-based routing methods. The notion of 'geographic spread' based cost minimization proposed in [19] seems to be appropriate. However, it needs to be augmented with the data flow characteristics for effective use in multi-service networks. Consider, for instance, 2 sources  $s_1$  and  $s_2$  in a configuration. A case where high bandwidth video data flows from  $s_1$  and low bandwidth audio data flows from  $s_2$  is different from a case where only audio data flows from  $s_1$  and  $s_2$  each. In the former case, the paths from  $s_1$  and  $s_2$  to destinations are likely

to be accentuated more towards connecting  $s_1$  to destinations with less number of hops, while in the latter case, these paths are likely to be evenly spread out, in the physical topology.

## 5.2 End-to-end delay constraints

The architecture specification  $\mathcal{A}$  may also influence the setting up of paths with end-to-end delay guarantees. This is because an architecture restricts the selectable set of paths to only a subset of paths feasible in the physical topology.

Consider a destination  $d$  joining a channel configuration connecting to sources  $s_1$  and  $s_2$ . In CBT architecture, the delay condition is:

$$str\_del(x, SMP(x)) + str\_del(SMP(x), d) \leq del_q(x, d)|_{x=s_1, s_2},$$

where  $SMP(s_1)$  and  $SMP(s_2)$  may be any of the nodes in the ‘core’-rooted tree. In PIM architecture, the above condition needs to be evaluated for placement of  $SMP(s_1)$  and  $SMP(s_2)$  in the RP node that serves  $d$ . Suppose the above condition does not hold. With CBT architecture,  $d$  cannot connect to the channel. The PIM architecture however allows  $d$  to bypass the RP node and set up a source-rooted tree to  $s_1$  and  $s_2$  each that meets the delay condition (thus, the RP-tree and source-rooted trees coexist). Since the URT architecture does not pre-designate any particular node to serve as SMP, it is possible for  $d$  to change the choice of SMP node for  $s_1$  and  $s_2$  such that the delay condition can be met. This reduces the probability of ‘connect’ failures.

In a modification to the CBT architecture [20], the ‘core’ can be dynamically split into many ‘core’ nodes if the path to one or more destinations from the current ‘core’ node does not meet the delay condition. The topological placement of such ‘dynamic core’ nodes needs to be carefully done because, in some extreme cases, the join of each destination can cause a re-assignment of all the ‘core’ nodes.

We now describe the protocol model that we have designed in this project, to allow systematic acquisition and use of source-destination configuration information by a routing algorithm. The latter may use this information to optimally exercise network-wide resource allocation control.

## 5.3 Routing control protocol

The functional view of routing algorithms underscores a canonical protocol model that defines the decentralized flow-related information structures  $\mathcal{R}$  maintained by

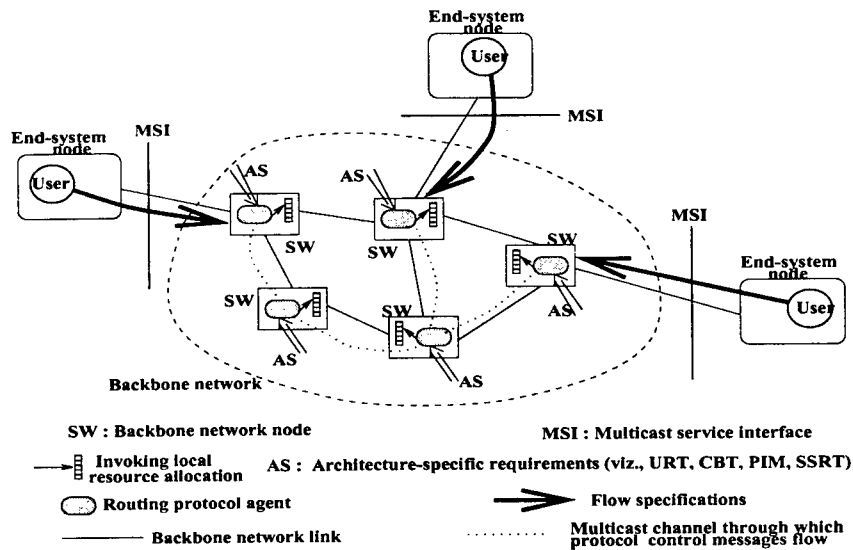


Figure 13: Protocol model for multicast path set up

'agents' executing at various nodes in the connectivity configuration  $\mathcal{C}$  and the routing control messages exchanged between these 'agents' to access the flow information. See Figure 13.

The messages propagate user-level flow specification and source-destination configuration information to the 'agents' in various nodes. The messages also carry information on resource allocations at various nodes/links and the overhead and delay incurred on the 'native connections' over these nodes/links. Representative types of messages include:

- 'search' messages to locate SMP nodes that can support the required flow from/to user;
- 'resource check' messages to ascertain the availability of resources and the end-to-end delay guarantees along downward path segments to support a flow;
- 'path setup' messages to allocate resources at nodes in the path through a given node;
- 'stream tap' messages to locate a stream flowing along upward paths and bring it to a given node;
- 'path tear down' messages to de-allocate resources along chosen path segments;
- 'stream remove' messages to delete a stream flowing through a node and along its upward path segment up to a point where the stream needs to flow along other branches of the tree.

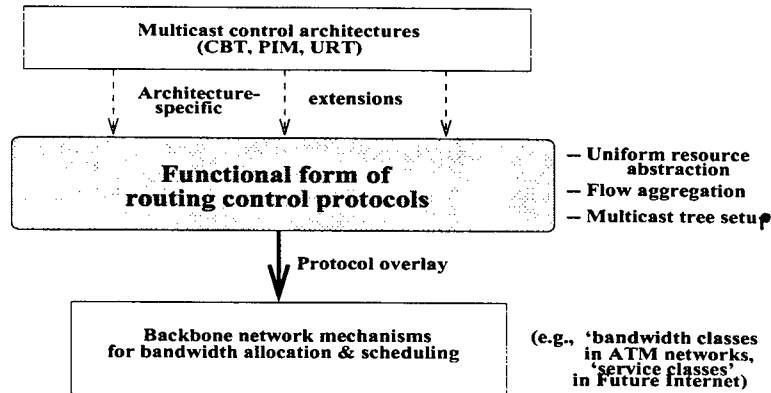


Figure 14: A 'protocol-oriented' view of resource control system

During execution, the protocol may root a tree at any chosen node to propagate these control messages to all other nodes in the path segments.

Details of the protocol model in terms of canonical information structures maintained at 'agent' nodes and the control message flows across nodes to access these information may be found in [21]. The model is closely related to the 'Resource ReserVation Protocol' (RSVP) [16] and the 'Internet Stream Protocol' (ST-II) [8] in light of their scope towards multi-service applications.

Our design philosophy has been that the protocol structure is: i) canonical and architecture-independent (e.g., 'stream filter' mechanism should be the same for both CBT and URT architectures), and ii) open-ended to allow architecture-specific extensions (e.g., the use of a 'unicast protocol' to locate the 'core' node in CBT and a 'cache' of potential access point node addresses in URT). See Figure 14. The various aspects of our protocol model reflect this philosophy.

#### 5.4 Simulation studies

The protocol model was studied by extensive simulations of routing algorithms that set up cost optimal paths, under the URT architecture. Large sized topologies were simulated (50-100 nodes with 'node degree' of 2-5), with dynamic application configurations. Two types of metrics were analyzed:

- The 'time overhead' and 'control message overhead' incurred network-wide to effect configuration changes;
- The 'degree of path sharing' across various data flows, as achievable by the protocol.

The 'time overhead' may indicate the latency experienced by joining user entities; this may translate to, for example, a 'TV channel switching time' in digital TV receivers. The 'message overhead' may indicate a one-time network-incurred cost of effecting the join of a user to a configuration. The 'degree of sharing' may indicate the network-wide savings possible in flow-related resource allocations for a given configuration. An analysis of these metrics for a variety of network topologies and application configurations indicates a feasibility of the protocol model for use in multicast networks. See [21, 22] for details of these simulation studies.

The functional elements and message structures defined for the protocol are common across the CBT, PIM and URT architectures. Architecture-specific extensions can be incrementally added in the target implementation of a routing algorithm. These details may also be found in [21]. In a degenerate form that does not employ flow aggregation, the protocol can also support the SSRT architecture.

## **6 Implementation study over ATM networks**

We have studied the functional model of multicast routing algorithms on an ATM network consisting of 5 nodes (Fore and GTE switches) and 3 SUN-Sparc workstations. These workstations implement the source and destination entities.

### **6.1 Routing protocol overlay**

Since the native ATM switches do not support our functional model, we placed the protocol agents in 'logically extended switches' (LES), implemented on (additional) workstations attached to native ATM switches. LES realizes the protocol overlay, with a convergence layer for ATM networks (see Figure 15). The VP/VC set up and VP level bandwidth allocation functions provided in the native ATM switches are available to the agents in LES through a programming interface. The physical topology of backbone network is basically a set of VP links interconnecting the various switches. Data flow paths and control message paths are realized through separate VCs multiplexed on common VP links (a VP link is a unidirectional channel between a pair of ATM switches).

Data paths generated by the routing algorithm are specified in terms of bandwidth needs on each VP link and the filters placed along each VP link. The logical topology of data paths is then embedded onto the ATM network VC/VPs as follows. The estimated bandwidth needs for a VP link are transcribed into the native switch bandwidth allocation for VP links. A distinct VC is assigned to each stream in the

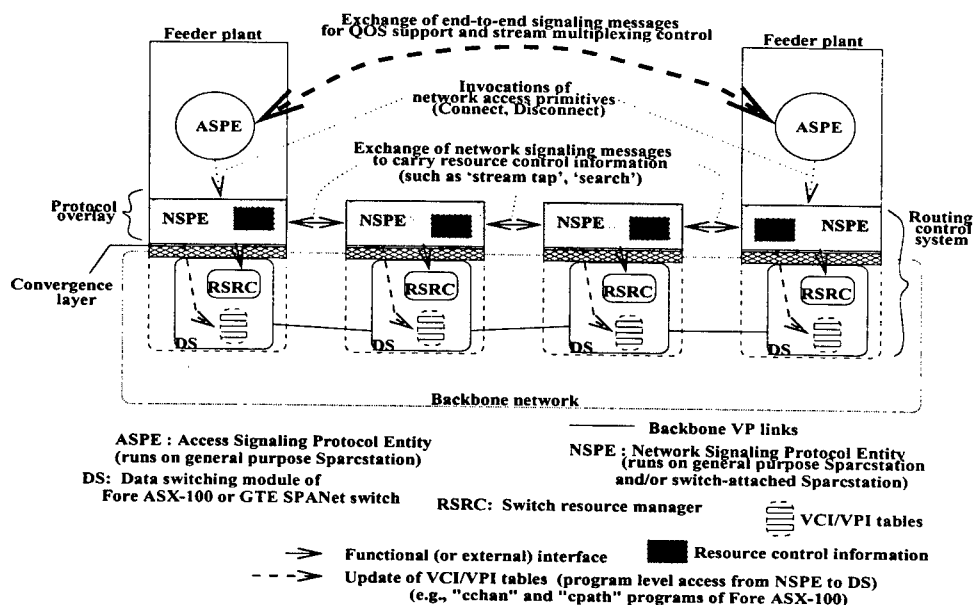


Figure 15: An implementation structure for 'network level signaling' to support resource allocations in ATM networks

application (as suggested in [23]). Multiple VCs share one or more common VP links, as allowed by the multicast control architecture<sup>3</sup>. A stream filter along an outgoing VP link is realized by simply not including the corresponding VCI-VPI mapping entry in the switch tables for this VP link. A bidirectional path segment is realized with 2 unidirectional VC/VPs since the native ATM switches support only unidirectional channels.

The 'statistical multiplexing' of VCs over a VP based on 'bandwidth classes', as allowed by ATM switches, can be exploited in a coarse manner by mapping the flow rate  $q$  into an appropriate 'bandwidth class' and mapping the  $\Delta$  to one of the allowed 'cell loss priority' values. Further studies are required to establish a tighter linkage between our resource model and the ATM 'network service' offerings.

## 6.2 'native ATM signaling' support

Since the 'native signaling' protocols employed in FORE and GTE switches set up only SSRT paths using SVCs — 'switched virtual circuits', we resorted to the use of PVCs — 'permanent virtual circuits' — to create shared multicast VP links.

<sup>3</sup>Bandwidth savings due to link sharing across bursty flows could not be determined from this testbed, since the current version of ATM switches allocate the entire bandwidth for streams in the absence of contention and we could not generate enough traffic to saturate the switch capacity of 625 *mbps*. However, a separate study using ATM 'cell traffic analyzer' equipments confirm that bandwidth savings are possible due to 'statistical interleaving' of bursty streams.

Here, bandwidth estimates for aggregated flows are computed by LES nodes, which are then used in the setting up of the required PVCs. In this context, it may be noted that works are being pursued elsewhere to support 'shared trees' over ATM networks [24]. A spin-off of these works is to integrate flow management into ATM connectivity setup protocols and provide signaling support functions therein.

### 6.3 Application configurations

Two distinct application configurations are studied: one with 4 streams (2 video and 2 audio) and 4 destinations spread out across 3 workstations and the other with 4 streams (2 text and 2 graphics) and 2 destinations spread out across 2 workstations. The routing algorithm employs 'incremental path setup' as the cost constraint, and incorporates the architectural requirements of CBT, PIM or URT, as the case may be. The algorithm generates paths in each of these architectures, specified in terms of bandwidth needs on each VP link and the filters placed along each VP link. Stream delay information is also generated by the algorithm at each LES in a path<sup>4</sup>. Our experiments on the above system environment indicate that the time to complete the 'join' activity of a user is less than 20 msec.

The total number of distinct VP links required network-wide to support both the application configurations (fixed cost overhead) under the URT, SSRT, CBT and PIM architectures are found to be 20, 36, 28, and 30 respectively. This indicates the extent of resource savings possible with the URT architecture over other existing architectures.

As a qualitative note, our functional model of routing algorithms allowed an easier realization of various architectures and cost constraints. In fact, that it would have been impossible to implement multicast routing for multi-service networks with a traditional 'monolithic' view of routing algorithms is not an over-statement.

## 7 Study of multicasting over heterogeneous networks

We employed the URT model as a reference multicast architecture in this study. The results of the study can however be extrapolated to the CBT and PIM architectures as well.

To allow the mapping of MCL functions onto specific backbone networks, we adopted a network-wide *logical addressing* of URT channels that allows establishing local

---

<sup>4</sup>Stream delay information is of less consequence in this small testbed under study, except for validating the algorithm functionality of enforcing delay control.



bindings between a logical address and the information on network-specific routing of data over switches and links. For instance, the ‘communication division’ of Rome Laboratory may subscribe to a logical address that may be used by the routing system to deliver packets at various workstations belonging to this division. The logical addressing allows grouping of selected destinations to overlay different ‘virtual networks’ on a base level multicast channel (e.g., private discussion groups in a conference). As a demonstration of this architectural feature, we designed strategies for the embedding of the URT model on sample backbone networks: interconnected LANs and ‘SMDS networks’ (Switched Multimegabit Data Service)

### 7.1 Semantics of logical addresses

A channel  $\mathcal{G}$  is bound to a unique logical address  $addr$  that may be used as a network-wide index for the underlying multicast path. The  $addr$  may be specified at the time  $\mathcal{G}$  is created by a user level management entity, and may be inherited by every router node of  $\mathcal{G}$ . A user entity  $U$  may connect to  $\mathcal{G}$  by specifying  $addr$ . A multicast routing algorithm uses  $addr$  as a key to locate a SMP in  $\mathcal{G}$  for creating a path segment between  $U$  and the SMP. A source prefixes each data packet sent across the UTL-MCL interface with a label in the header containing  $addr$ , to enable the routing system in the network determine the path through which the packet is to traverse.

A logical address  $addr$  binds only to a multicast path, but not (directly) to the entities participating in data exchanges over this path. This semantics allows route processing based on  $addr$  to be an integral part of multicast implementation on backbone networks. In contrast, existing group addressing schemes directly bind the user level entities, as a group, to a single address, such as the ‘host group address’ for inter-machine communications over LANs and Internet [2, 25]. As a case of logical addressing, the 32-bit unstructured format of ‘IP Class-D’ address [1] is a suitable candidate for assignment to a URT channel, that employs ‘IP multicast’ based link-to-link data connectivity among the component gateway nodes. Figure 16 illustrates ‘network-oriented’ and ‘user-oriented’ addressing schemes employed in data transport networks.

The UTL entities at various access nodes need to manage the shared logical address space  $\{addr\}$  for allocation and de-allocation of addresses to multicast paths and bind these addresses to application level names of communicating objects implemented by the UTL entities. Such a function may be provided as part of a systemwide name service. The evolving standards such X.121 ‘subscriber addressing’ [26] and SMDS

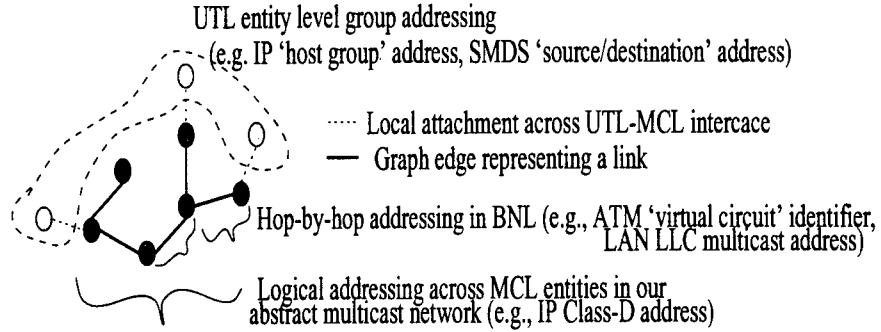


Figure 16: Path addressing in various layers of multicast network architecture

'group addressing' [27] may be used to assign application level names.

## 7.2 Packet forwarding

Consider a set of URT channels that each pass through a switch  $S$ . Each of these channels may have path segments to one or more neighbors of  $S$ . This information is maintained by  $S$  in a *routing table*, with each entry in the table corresponding to a channel. For a channel bound to address  $addr$ , the corresponding table entry provides a mapping of the form

$$addr \rightarrow \{p_{id_1}, p_{id_2}, \dots, p_{id_r}\}$$

with  $1 \leq r \leq K$ , where  $p_{id_r}$  is the port through which  $S$  has an edge to a neighbor  $S^r$  and  $K$  is the number of ports of  $S$  in the physical topology. As can be seen,  $S$  does not have complete information about a route. Instead, a decentralized structure of the routing information is employed namely,  $S$  only knows through which of its ports (or, links) the entities connected to  $addr$  can be reached. This technique is known as *late binding* of routes.

With *source-routing*, the network level agent of a source entity residing in a switch  $S_0$  maintains the routing information for the entire tree rooted at  $S_0$ . In general, the subtree rooted at a switch  $S$  is representable as an ordered list of port ids in the form

$$list(addr) = [(p_{id_1}, list_1), (p_{id_2}, list_2), \dots, (p_{id_r}, list_r)],$$

where  $list_j$  represents a subtree rooted at switch  $S^j$  to which  $S$  has a branch through  $p_{id_j}$ . To multicast a packet,  $S$  sends the packet through each  $p_{id_j}$  affixing

$list_j$  in the packet header. On receiving the packet,  $S^j$  uses  $list_j$  as the tree to multicast the packet through. This type of routing may reduce the per-packet processing overhead by avoiding the routing table lookup.

The source-routing may be useful for 'closed user group' connections (e.g., small conference sessions, replicated database access) and unicast channels where the number of leaves in a tree is not large, say  $\leq 10$ , and hence the header overhead to carry the port list can be a small fraction of the channel bandwidth. For example, a 4-bit field for  $pid_r$  ( $K \leq 16$ ), a 4-bit delimiter field for various branches at a node,  $\frac{r}{K} = 0.1$  and up to 4 hops along the tree can require a header approximately of size 20 bytes to carry the list of ports. This consumes about 4% of the bandwidth allocated for a path, assuming 512-byte packets. This overhead when compared to that caused by a 4-byte field to carry an  $addr$  in the header may be acceptable in light of the packet processing overhead saved on a routing table lookup for every data packet. With 'open user group' connections (e.g., large conference sessions, broadcast TV and audio) however, the number of leaves may be large, say 100. So<sup>5</sup> the header size can be as high as 100 bytes using the earlier example with  $\frac{r}{K} = 0.5$ , which consumes about 20% of the bandwidth. Such an overhead may not be acceptable when the scalability of implementation with the size of application configurations is important.

Thus the advantage of source-routing over late binding of routes in the form of increased packet switching efficiency is limited to only when the branching factor of the tree, indicated by  $\frac{r}{K}$ , is small (as in connections with unicast entities and some 'closed user groups') because of the size of the list carried in the packet. So source-routing may be resorted to only as an alternative to late binding of routes, rather than as a rule.

Overall, the network support for both routing modes as part of logical addressing is desirable for multi-service networks.

### 7.3 Implementation strategies for IP LANs

Consider a URT channel that spans across a LAN  $X$  and an IP-based internetwork of LANs  $Y$ . The MCL in the gateway  $G$  that interconnects  $X$  and  $Y$  is basically a guest level implementation in which the 'logical link control' (LLC) layer of  $X$  and the IP layer of  $Y$  are viewed as providing subnet-specific multicast facilities across the 'LAN hosts' residing in  $X$  and  $Y$  respectively. So the MCL maps the logical

<sup>5</sup> A metropolitan network configuration can attach 1000 TV receivers to a network node (through fiber feeder plants). With 100,000 TV receivers tuned to a channel at any time, a multicast tree may have 100 leaves.

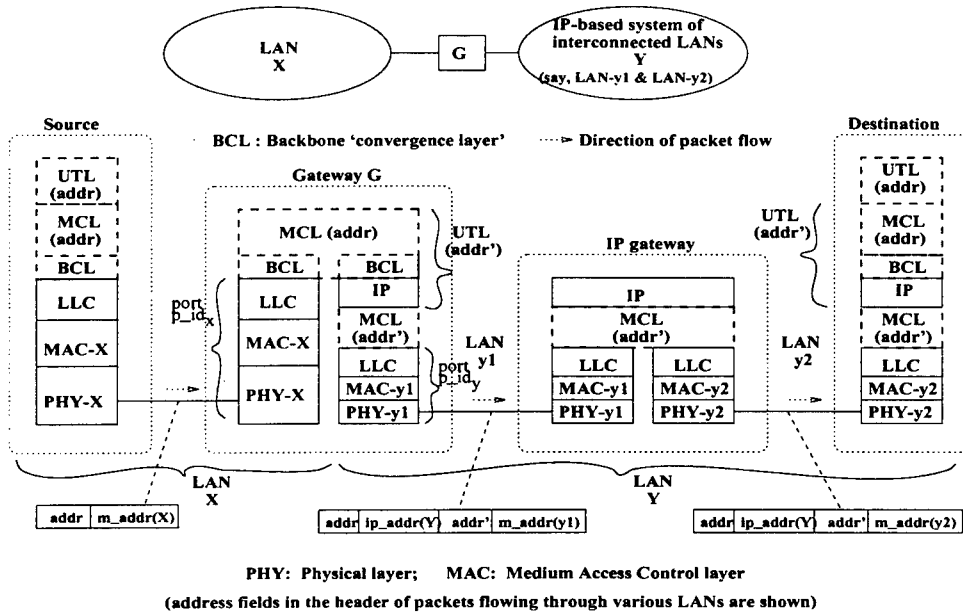


Figure 17: Multicasting across interconnection to IP LANs

address  $addr$  associated with the channel to a LLC multicast address  $m\_addr_X$  of hosts connected to  $X$  and a IP level 'group address'  $HG_Y$  of hosts connected to  $Y$ . This mapping may be represented as:

$$addr \rightarrow \{ (p\_id_X, (NET\_MCAST, (m\_addr_X, \dots))), (p\_id_Y, (NET\_MCAST, (HG_Y, \dots))) \},$$

where  $p\_id_X$  and  $p\_id_Y$  are the ports through which  $G$  activates the LLC layer functions of LAN  $X$  and the IP layer functions of LAN  $Y$  respectively, and NET\_MCAST is a flag indicating the availability of 'native multicast' facility in the backbone network or otherwise. With this type of binding in  $G$ , each packet received over  $X$  with  $addr$  as its label is forwarded over  $Y$  at the 'host group' address  $HG_Y$  using the IP multicast protocol available within  $Y$  (the latter may possibly use the underlying LAN broadcast facility). At a destination, a packet arriving in the IP layer at  $HG_Y$  is passed on to the MCL residing above for delivery at various destinations connected to  $Y$ . See Figure 17.

Since the IP LAN  $Y$  may be viewed as a system of interconnected LANs in itself, another instantiation of MCL may be interposed beneath the IP layer but above the LLC layer of  $Y$ . With this splicing of MCL functions into  $Y$ , the IP 'host group' address  $HG_Y$  may be bound to a logical address  $addr'$  for multicasting packets within  $Y$  using the URT model (i.e.,  $HG_Y$  appears as a UTL address in  $Y$ ). So  $Y$  forms a

separate multicast domain, with the mapping between  $addr'$  and  $HG_Y$  done by the MCL functions at nodes in this domain.

Thus an implementation of our multicast model extending across LANs  $X$  and  $Y$  consists of both guest layering and splicing of MCL functions, represented as:

$$addr \rightarrow \{ (p\_id_X, (NET\_MCAST, (m\_addr_X, \dots))), \\ (p\_id_Y, (NET\_MCAST, ((HG_Y \rightarrow addr'), \dots))) \}.$$

This may be desirable when multipoint-to-multipoint communication support needs to be provided within the IP LAN  $Y$  as well. As can be seen, the URT model can be recursively applied at various levels in a LAN interconnection architecture, with each level of recursion mapping to nodes connected at this level of addressing.

#### 7.4 Implementation strategies for SMDS networks

The network providing the SMDS may consist of one or more MAN Switching Systems (MSS) interconnected with one another across a wide area and/or metropolitan area. An MSS can be a high speed data switching node or a high speed LAN/MAN, such as FDDI and DQDB. The MSSs and the interconnection between them constitute the backbone network.

A user sees SMDS as providing a 'connection-less' data service. So each data frame exchanged across the service interface carries the full source and destination address in its header. The SMDS protocol requires the individual address of a source entity and a group address for destination entities (based on E.164 ISDN numbering) in each SMDS frame. This entity level addressing is at a higher level than our logical addressing of multicast channels. So a sublayer in the SMDS-network interface (SNI) needs to establish the mapping between these two levels of addressing.

The protocol architecture in [27] suggests a framework for interconnecting the MSSs to export the required data service to subscribers. We adopt this architecture to embed the URT model into the SMDS network for extending its multicast capabilities. The architecture consists of three layers of functions: Network Service and Control (NSC) layer that maps SMDS access requests to operations on multicast channels, Routing and Relaying (RR) layer that provides a subnet-independent realization of channel operations with appropriate protocols, and Component Network Convergence (CNC) layer that elevates the MSS specific functions to a uniform level for use by the RR layer. Thus the NSC layer implements the SMDS interface to multicast channels, while the RR and CNC layers provide an implementation of multicast control functions. Accordingly, the URT architecture needs to be grafted

at the NSC-RR layer interface of the SMDS network architecture, with the MCL and UTL functions embedded into the RR layer and NSC layer respectively; the CNC layer corresponds to an adaptation layer that is interposed between the MCL and BNL.

To map a SMDS level group address  $g\_nm$  of destinations to logical addresses for use by the underlying RR and CNC layers, a NSC entity maintains two types of information:

1.  $g\_nm \rightarrow \{ent\_addr\}$  to bind  $g\_nm$  to SMDS level addresses  $\{ent\_addr\}$  of entities attached to the local SNI;
2.  $g\_nm \rightarrow addr$  to bind  $g\_nm$  to a transport level logical address  $addr$  for multicasting.

An entity  $U$  that wishes to send and/or receive data over  $g\_nm$  interacts with its NSC entity across the SNI along the control plane to include its SMDS level address in the binding information (1). When  $U$  does not wish to send and/or receive any more data, it interacts with the NSC to remove its address from the binding information (1). The binding given by (2) can either be static whereby  $addr$  can be derived from the naming structure used for  $g\_nm$  or be dynamic whereby  $addr$  may be generated from a pool of logical addresses maintained by the NSC layer.

Each data frame exchanged across the SNI for multicasting includes, in its header, the individual address of the source and the group address of the destinations. So a NSC entity receiving a frame  $F$  across the local SNI for sending to destinations at group address  $g\_nm$  extracts  $addr$  from the binding information (2) and sends  $F$  to the RR layer to multicast  $F$  to the peer NSC entities at address  $addr$ . The RR layer prepends  $addr$  to the header and passes it down to the CNC layer for subnet-specific processing of the route information generated from  $addr$  and dispatching  $F$  through the component subnets towards destinations. See Figure 18 as an illustration. When a peer NSC entity receives  $F$  from the RR layer, it uses the  $g\_nm$  carried in the header of  $F$  to extract  $\{ent\_addr\}$  from the binding information (1) and delivers  $F$  across the local SNI to all the listed SMDS entities. The source address is not used for sending/receiving  $F$ , but may be used by the SMDS level protocols that generate and consume  $F$ .

The aforementioned approach to realize SMDS aligns well with the established approaches to LAN-MAN interconnections [28].

We believe that other multicast architectures, viz., SSRT, CBT and PIM, can also

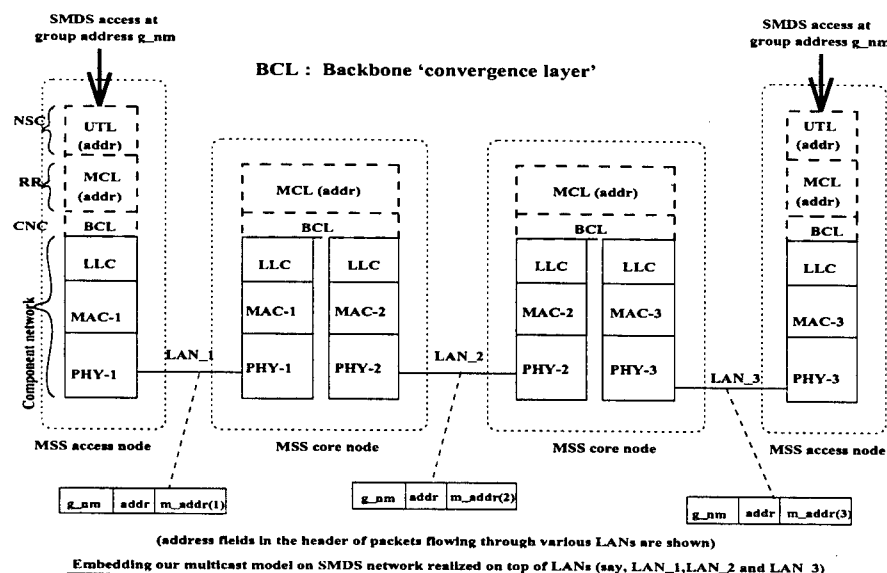


Figure 18: Overlaying URT-based multicast model on SMDS networks

be likewise realized on interconnected LANs and SMDS networks<sup>6</sup>.

## 8 Development of end-system support mechanisms

This section deals with providing the mechanisms for displaying multiple video windows and supporting other application devices on a workstation. The formulation of end-system mechanisms was merely to demonstrate the feasibility of multicast application implementations and to offer an evolutionary path towards such implementations.

The work in this category pertains to the design of: i) logical address based 'device grouping' mechanism for merging/splitting of data streams in a workstation, and ii) node software structure for multicast transport and/or generation/delivery of data units, as described below.

<sup>6</sup>Consider, say, a subtree of the 'core'-rooted tree with the 'core' at node  $T$ , spanning the nodes  $\{x, y, z\}$ . Assume that the subtree has a bidirectional path segment between  $x$  and  $y$ , and a unidirectional path segment from  $x$  to  $z$ . It is possible that this subtree resides on a distinct subnetwork that internally employs another instance of the CBT architecture, wherein  $x$  and  $y$  act as source entities sending their data each towards a 'core' node, say,  $T'$ , for onward distribution to destination entities  $y, z$  and  $x, z$  respectively over a tree rooted at  $T'$ , using the native routing protocols supported in this subnetwork.

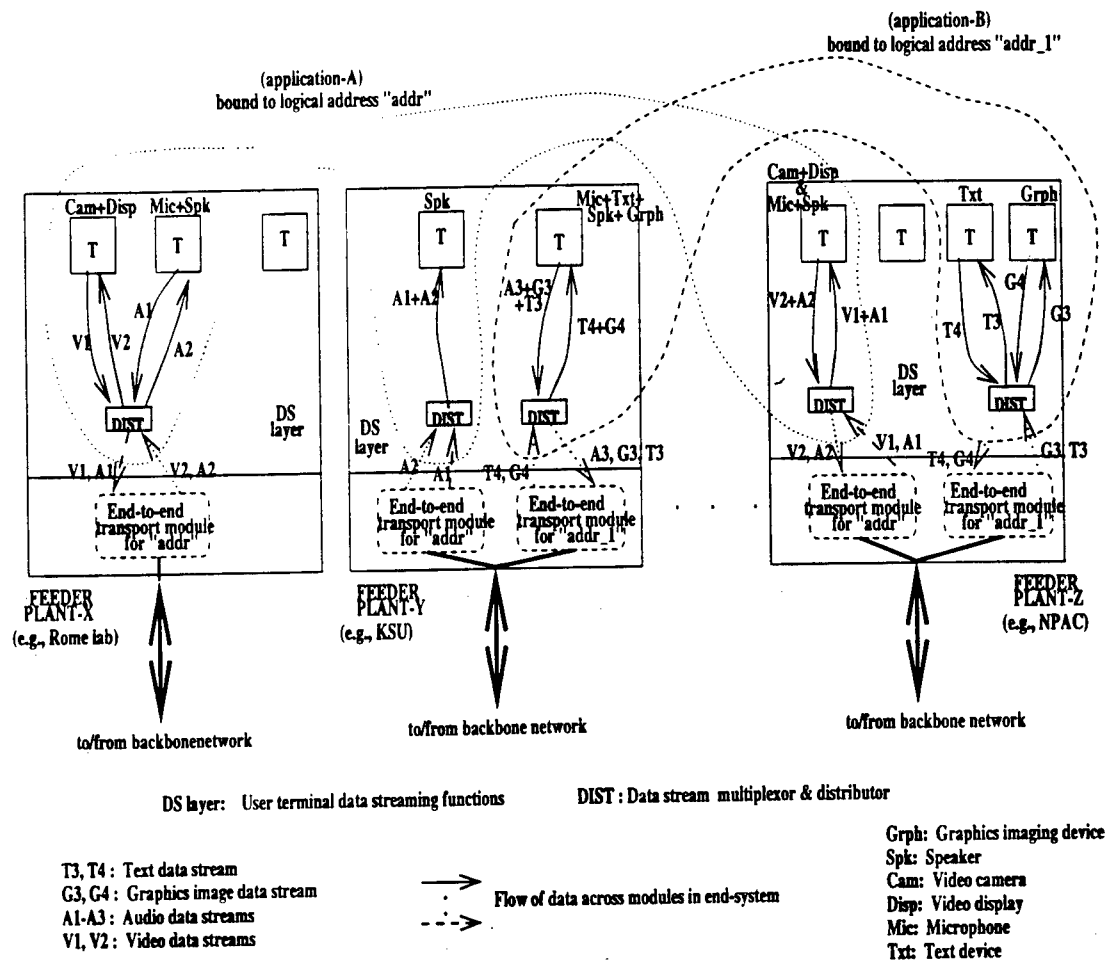


Figure 19: Illustration of feeder plant mechanisms

## 8.1 Grouping of application-level devices

A workstation mimicks an *access feeder plant* in a large network of subscriber terminals. A feeder plant maps the various subscriber terminals, viz., devices, that participate in an application onto a single 'logical device' for the purpose of presenting/fielding data to/from the underlying multicast network. This is achieved by logical address based grouping of physical devices. See Figure 19.

The logical address based grouping of devices in a feeder plant allows end-to-end multiplexing of data streams generated by various devices that participate in a given application. This allows the QOS management

for the multiplexed data streams in the following forms:

- Estimating the data rates of combined streams from that of the component streams;
- Handling of delay and loss tolerance attributes by stream-specific scheduling of data transport in end-systems.

Developing specific QOS management policies is itself outside the scope of the contract with Rome Laboratory.



With the use of logical address based device grouping, the network access interface allows addition and deletion of streams to an on-going data flow under a given logical address. Formulation of primitives for use in feeder plants is itself outside the scope of this project.

## **8.2 Workstation software for multicast transport**

A generalized structure of node software to allow the multicast forwarding of data packets — both video and non-video data — has been designed and implemented on SUN/Solaris workstations (see Figure 20). The software structure basically uses a shared packet memory between the network receiver, source device, network transmitter and destination device modules. The main ideas in the design are:

- Elimination of data copying from the address space of receiver/source modules to that of the transmitter/destination module by placing data packets in the shared memory;
- Employing less expensive inter-module synchronization mechanisms in the form of:

Software-implemented mutual exclusion and ordering among the per-packet processing of various modules;

Allowing a high degree of concurrency among executions of various modules during packet processing;

We employed a 'bounded buffer' based synchronization mechanism.

The above aspects are in general employed in implementations of high speed network software.

## **8.3 Connecting application devices**

This pertains to the distribution of data packets from various source devices (such as video cameras and microphones) attached to workstations over the multicast network to destination devices (such as video displays and speakers) on selected workstations. From the network perspective, the application devices in a workstation acting as a node appear as connected to the data transport modules through pseudo-ports that have distinct names and are assignable as distinct entries in the network routing table along with real network ports in the node. This allows a uniform treatment of data movement to/from the network and across the application-network interface.

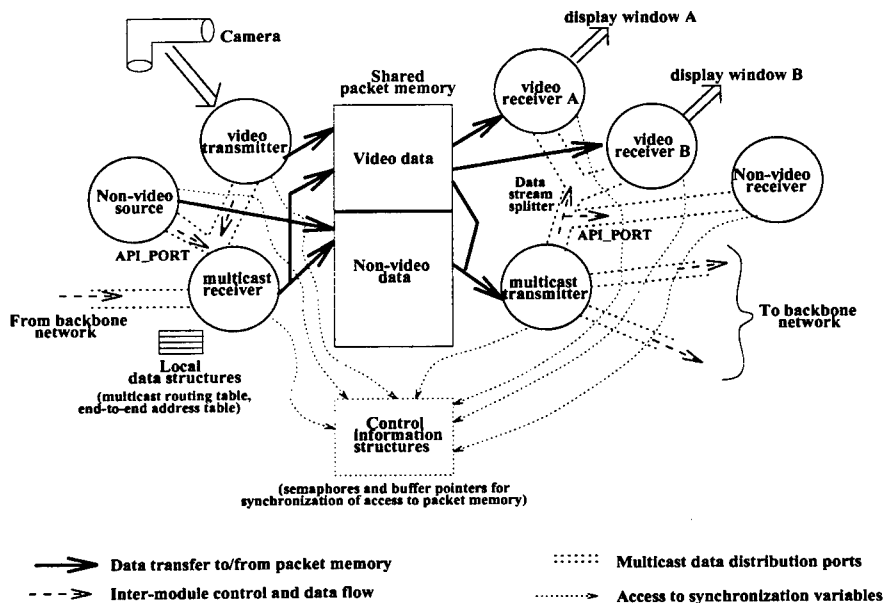


Figure 20: Multicast end-system software modules in a workstation

The software incorporating this structure assigns distinct end-to-end addresses for various devices which can be used to index to appropriate portions of packet memory and the associated information elements. The software handling application device modules then consists of generic *stub* interface and a device-specific packet handler. The stub interface deals primarily with:

- Mapping of device level addresses to control information elements and packet memory;
- Synchronization of device access to shared packet memory and control information elements (such as pointers to the start and final slots in a 'bounded buffer'), with the network receiver and transmitter modules.

Note that the use of generic transport level stubs is a standard method for connecting a variety of devices to any underlying network.

The node software incorporating the inter-module synchronization procedures and transport level stubs has been implemented on the SUN workstations. Care has also been taken to isolate backbone-network specifics from the software structure, which allows adopting the same software for both LANs and ATM networks with minimum changes in functionality.

The end-to-end delivery performance of the node software was measured on both 10 *mbps* Ethernet LANs and 155 *mbps* ATM networks. We achieved a throughput

rate of about 7 *mbps* over UDP/IP/Ethernet (where a packet was repeatedly sent over multiple UDP paths to realize the multicast), about 8.5 *mbps* over 'IP multi-cast'/Ethernet, and about 12 *mbps* on a network of Fore ATM switches. See [29] for the details of performance engineering techniques employed in the node software. We did not, however, embark on implementations aimed at maximizing the throughput rate and providing end-system level QOS support, since these are outside the scope of the project contract with Rome Laboratory.

## References

- [1] U. Black. **The Network Layer: Internetworking.** In chapter 6, *OSI: A Model for Computer Communication Standards*, Prentice-Hall Publ. Co., 1991.
- [2] D. R. Cheriton and S. E. Deering. **Host Groups: A Multicast Extension for Datagram Internetworks.** In *9th Data Communications Symposium*, ACM SIGCOMM, pp.172-179, Sept. 1985.
- [3] S. E. Deering and D. R. Cheriton. **Multicast Routing in Datagram Internetworks and Extended LANs.** In *ACM Transactions on Computer Systems*, Vol.8, No.2, pp.85-110, May 1990.
- [4] B. M. Waxman. **Routing of Multipoint Connections,** In *IEEE Journal on Selected Areas in Communications*, Vol.SAC-6, No.9, pp.1617-1622, Dec. 1988.
- [5] S. Shenker. **Fundamental Design Issues for the Future Internet.** In *IEEE Journal on Selected Areas in Communications*, Special Issue on *Global Internets*, vol.13, no.7, pp.1176-1188, Sept. 1995.
- [6] J. N. Chiappa. **Evolutionary Possibilities for the Internetwork Layer.** In Part IV, *IPng: Internet Protocol Next Generation*, Addison-Wesley Publ. Co., 1996.
- [7] D. D. Clark. **Foreward.** *IPng: Internet Protocol Next Generation*, Addison-Wesley Publ. Co., 1996.
- [8] C. Topolcic. **Experimental Internet Stream Protocol, version 2 (ST-II).** In *RFC 1190*, CIP Working Group, Oct. 1990.
- [9] D. Waitzman, et al. **Distance Vector Multicast Routing Protocol** In RFC 1075, 1988.
- [10] T. Ballardie, P. Francis and J. Crowcraft. **Core Based Trees (CBT): An Architecture for Scalable Inter-Domain Multicast Routing.** In *Proc. Comm. Architectures, Protocols and Applications*, ACM SIGCOMM, pp.85-95, Sept. 1993.
- [11] S. Deering, D. Estrin, D. Farinacci and V. Jacobson. **An Architecture for Wide-Area Multicast Routing.** In *Proc. Comm. Architectures, Protocols and Applications*, ACM SIGCOMM, Sept. 1994.
- [12] K. Ravindran. **A Flexible Network Architecture for Data Multicasting in High Speed 'Multi-service Networks'.** In *IEEE Journal on Selected Areas in Communications*, Special Issue on *Global Internets*, vol.13, no.8, pp.1426-1444, Oct. 1995.

- [13] K. Ravindran. **Communication Channels for Data Multicasting in 'Multi-service Networks'**. In *IFIP conf. on Network Protocols, Architectures and Applications*, Madras (India), Dec. 1994.
- [14] K. Ravindran. **Network Abstractions and Resource Allocation Models for Data Multicasting in 'Multi-service Networks'**. In *Tech. Report 93-7*, Dept. of Computing and Information Sciences, Kansas State University, Aug. 1995.
- [15] T. J. Gong and K. Ravindran. **Cost Analysis of Data Transport Architectures for Multicasting in 'Multi-service Networks'**. In *Tech. Report*, Dept. of Computing and Information Sciences, Kansas State University, Mar. 1995.
- [16] L. Zhang, S. E. Deering, D. Estrin, S. Shenker and D. Zappala. **RSVP: A New Resource ReSerVation Protocol** In *Network*, IEEE Communications Society, pp.8-18, Sept. 1993.
- [17] V. P. Kompella, J. C. Pasquale and G. C. Polyzos **Multicasting for Multimedia Applications**. In *proc. Computer Communications, INFOCOM'92*, Italy, 1992.
- [18] D. C. Verma and P. M. Gopal. **Routing Reserved Bandwidth Multi-point Connections**. In *Proc. Comm. Architectures, Protocols and Applications*, ACM SIGCOMM, pp.96-105, Sept. 1993.
- [19] J. Kadirire. **Minimizing Packet Copies in Multicasting by Exploiting Geographic Spread**. In *Computer Communication Review*, ACM SIGCOMM, vol.24, no.3, pp.47-62, July 1994.
- [20] Y. C. Chang, Z. Y. Shae, and M. H. W. LeMair. **Multiparty Video Conferencing using IP Multicast**. In *IS & E / SPIE Symp. on Electronic Imaging: Science and Technology*, 1996.
- [21] K. Ravindran and T. J. Gong. **Flow and QOS based Routing Control Protocols for Data Multicasting in 'Multi-service Networks'**. In *Tech. Report*, Dept. of Computing and Information Sciences, Kansas State University, Jan. 1996.
- [22] K. Ravindran and T. J. Gong. **Path Sharing Across Multi-source Flows in Multicast Data Transport**. In *Tech. Report*, Dept. of Computing and Information Sciences, Kansas State University, July 1996.
- [23] A. Demirtjis, B. Edwards, B. Braden, S. Berson, M. P. Maher, and A. Mankin. **RSVP and ATM Signaling**. In *ATM Forum/96-0258*, MPOA Sub-working Group, ATM Forum Technical Committee, Jan. 1996.
- [24] M. Grossglauser and K. K. Ramakrishnan. **SEAM: A Scheme for Scalable and Efficient ATM Multipoint-to-Multipoint Communication**. In *Tech. Report (draft)*, AT&T Research, Aug. 1996.

- [25] D. R. Cheriton and W.Zwaenopoel. **Distributed Process Groups in the V-Kernel.** In *ACM Transactions on Computer Systems*, Vol.3, No.2, pp.77-107, May 1985.
- [26] CCITT Recommendation X.121. **Network Addressing.** In *Data Communication Networks: Transmission, Signaling and Switching Requirements — Maintenance and Administration*, Vol.8.3, 1988.
- [27] BellCore. **Generic System Requirements in Support of Switched Multi-Megabit Data Service.** In *Technical Advisory*, TA-TSY-000772, Issue 2, March 1989.
- [28] A. Tantawy and M. Zitterbart. **A Scheme for High Performance LAN Interconnection Across Public MANs.** In *IEEE Jour. on Selected Areas in Communications*, Vol.11, No.8, Oct. 1993.
- [29] K. Ravindran, K. Bhat, T. J. Gong, K. Gould and D. Loguinov. **Performance Engineering of End-systems for High Bandwidth Multimedia Communications.** In *Tech. Report*, Dept. of Computing and Information Sciences, Kansas State University, Oct. 1995 (revised in April 1996).
- [30] S. Shenker and et al. **Network Service Element Specification Template.** In *RFC draft*, IETF WG on 'Integrated Services', Nov. 1995.
- [31] M. Macedonia and D. Brutzman. **MBone Provides Audio and Video Across the Internet.** In *IEEE Computer*, vol.27, no.4, pp.30-36, April 1994.

## **9 Technical deliverables**

The concrete deliverables were categorized under the various project activities, as described in sections 3-8. Detailed description of these deliverables may be found in the various technical reports submitted to Rome Laboratory POC ('point of contact') during various phases of the project.

### **9.1 Study of multicast network architectures**

A new architecture that employs 'unrooted tree'-based multicast channels was designed in this project. As part of this activity, the following phrases were completed:

1. A flow & QOS based cost model consisting of metrics for comparing different multicast architectures;
2. Results on bandwidth gains achievable in various architectures;
3. Analysis of the interactions between multicast routing algorithms and network architectures.

See [13, 14, 15] for the details on how the cost metrics are derived and estimated and on the experimental measurements of bandwidth consumptions.

### **9.2 Design of multicast routing control protocols**

A canonical protocol model was developed that allows flow & QOS specifications to be transcribed onto network internal mechanisms. The latter manifest in the form of identifying the functional elements in the network and defining the message space and information structures for use by multicast router nodes. Algorithmic techniques for decentralized resource allocation were formulated, and then evaluated by simulation. See [21, 22] for details of the protocol model and its evaluation studies.

### **9.3 Design of signaling system overlays on ATM networks**

Strategies for overlaying the routing control protocol on ATM networks were defined and implemented on the testbed consisting of Fore and GTE ATM switches. PVC-based techniques for the embedding of various multicast architectures (viz., SSRT, CBT, PIM and URT) were identified and evaluated. Details of these strategies and the evaluation results may be found partly in [12, 21].

#### **9.4 Interconnection of heterogeneous backbone networks**

Strategies for realizing a global multicast over interconnected heterogeneous networks were formulated, and then evaluated on interconnected LAN and SMDS backbone network architectures. These strategies were based on associating a global logical address to each multicast channel and then binding this address into the underlying native routing system of the backbone network. Details of these strategies may be found in [12].

#### **9.5 End-system software on workstations**

Structuring techniques for the end-system modules, based on logical grouping of application devices, were formulated and evaluated on SUN-sparc-5 workstations. Also, performance engineering techniques to maximize the end-to-end throughput rate were studied. Experimental results collected as part of this activity and analysis of these results may be found in [29].

#### **9.6 Software demonstration**

A multicast application involving the distribution of video, audio and graphics data was demonstrated on top of the ATM-based NYNET between Syracuse University and Rome Laboratory. The demonstration was presented to the Rome Laboratory technical staff in the 1st week of August 1996. The demonstration evidenced, in part, a successful deployment of the network mechanisms developed in this project. The aforementioned deliverables serve to fulfill the work items stipulated in the **Statement of Work** attached to the contract with Rome Laboratory. For interested readers, the technical reports may be obtained from the Rome Laboratory POC.

### **10 Project contributions to Air Force research mission**

The subject category of **Distributed Information Environment** under which this project was carried out deals with 'information connectivity' among large 'information repositories' using high speed networks. For example, large scale strategic and tactical information (e.g., terrain information about military installations and movements) may need to be distributed as imaging data across different Air Force Centers for dissemination purposes. As another example, business executives of a company or field commanders in a battlefield may engage in a teleconference session, involving video, audio and graphics data, to coordinate their activities. Our project



on transport level connectivity in networks may be viewed as providing a 'flexible communication backplane' among 'information repositories' to allow such large scale information transfers. Also, the project will allow smooth development of computer systems for information movement and dissemination at higher levels of 'information architectures'.

## 11 Future works

As the project work evolved and matured, quite a number of topics were identified that can be candidates for future work. These are:

1. Incorporation of 'receiver-initiated' style resource control protocols into a generic signaling system

What type of end-to-end protocol support is required to allow 'receiver-initiated' style protocols in the network, has been a topic of debate in the Internet community. Many RFCs have emerged in the past 2-3 years, but no consensus has yet been reached on a single set of reservation mechanisms that may satisfy the needs of a vast majority of applications.

2. Integration of flow management in ATM networks

What model of ATM level network connectivity is required in order to smoothly integrate flow management procedures onto the native QOS support mechanisms ? This problem has attracted the attention of many research organizations (see [23, 24], for instance).

3. Flow aggregation strategies for network connections

While there is a general agreement in the research community that substantial savings in communication resources is possible by sharing a network path across multiple data streams, there has not yet been any agreed set of mechanisms for aggregating these data flows into a single composite flow without compromising one or more of the transport functionalities. For instance, when a delay sensitive stream (such as video) is merged with a delay insensitive stream (such as text), the question that arises is: what is the delay sensitiveness of the composite stream ? Is it the minimum of the two sensitivity values (whereupon the text stream gets a 'free ride') or the maximum of the two values (whereupon the video stream 'suffers') ? The RFC [30] sheds some light into these questions.

4. Interconnection of 'multicast islands'

As networks increase in size (such as the exponential growth of the Internet), different regions of the network supporting different types of routing protocols becomes a major impediment to interconnecting them for data distribution purposes. The evolution of 'Mbone' is an example of a solution to this problem [31]. Though the use of 'point-to-point tunnels' to

interconnect the different 'IP multicast' regions is a start, it by no means is a systematic solution. We have been nurturing an idea of 'multi-point tunnels' using logical addresses as a generic solution to the problem.

The list of aforementioned topic areas of future works is by no means exhaustive.

An overall technology question in this context is: whether 'standardization of network subsystems' promotes the growth and evolution of networks, or is it a 'bottleneck' to the whole process of network evolution itself? A more appropriate stance may be to standardize the procedures for interconnecting the network subsystems, but not the subsystems themselves.

## 12 Scientific contributions

The research-oriented nature of the project work undertaken in this contract has spin-offs in many directions of basic and applied areas of Computing and Communication disciplines. We categorize them in four different topics.

### Network Engineering:

The project work exposed a clear-cut separation of the 'control plane' and 'data plane' functions in the network architecture. This allows focusing on the 'control' functions without intruding into the 'data flow' related functions, and vice versa — at least from a network designer's perspective. The functional separation aligns with the evolving architectures for B-ISDNs and the Internet.

### Software Engineering:

We resorted to an 'object-oriented structuring' of the multicast transport system to realize the underlying 'programmable network' theme. To exemplify, we designed a canonical set of routing control protocols that allow 'plug-n-play' of different types of multicast architectures. We adopted a similar 'plug-n-play' approach to supporting a variety of application devices by mapping them to a single instance of logical devices. We believe that the 'object-oriented' approach is necessary in the design of any large networking system with complex requirements.

### Distributed algorithms:

The management of decentralized information structures at router nodes is basically a distributed algorithm problem. In this exposition, the problem specifically manifests as finding cycle-free routes for a multicast channel ('safety' property) and guaranteed determination of routes ('liveness' property). The 'safety' and 'liveness' requirements need to be ensured in the presence of simultaneous join/leave of users to a multicast channel. For this purpose, 'concurrency control' techniques well-studied in distributed computing can be employed.

### **Complexity analysis of routing algorithms:**

Determining a cost optimal multicast path constitutes the 'steiner tree' problem, which has been shown to be NP-complete by Combinatorial Mathematics researchers. So many heuristics-based approximation algorithms have been proposed by the 'network routing' community. In fact, the 'incrementally cost optimal' and 'globally cost optimal' algorithms that we studied in this project are cases of approximation algorithms. With 'path sharing across multi-source flows' becoming an additional input parameter to cost-based routing algorithms, their computational complexity becomes an interesting research problem in the area of Combinatorial Computing.

The above expositions open up interesting avenues of fundamental research and/or technology innovations.

***MISSION  
OF  
AFRL/INFORMATION DIRECTORATE (IF)***

*The advancement and application of Information Systems Science  
and Technology to meet Air Force unique requirements for  
Information Dominance and its transition to aerospace systems to  
meet Air Force needs.*